CHEMNITZ UNIVERSITY OF TECHNOLOGY

Dynamic Graph Generation and an Asynchronous Parallel Bundle Method Motivated by Train Timetabling

DISSERTATION

submitted to

Department of Mathematics

at

Chemnitz University of Technology

in accordance with the requirements for the degree Dr. rer. nat.

Dipl.-Inf. Frank Fischer

Chemnitz, 2nd April 2013

Supervisor: Prof. Dr. Christoph Helmberg

URN: urn:nbn:de:bsz:ch1-qucosa-118358

Contents

1 Introduction

The planning and operation of railway systems is a complex and challenging task. Several aspects of operation have to be considered, usually in hierarchical form [20]:

- planning of the infrastructure network,
- line planning (which stations should be connected by which train routes),
- timetable construction (when do the trains run),
- track allocation, platform allocation (exactly which tracks should a train use, *e. g.*, in a station),
- rolling stock management (which locomotives should be used for which run),
- crew management (who should drive the train),
- real time management (*e. g.*, handle delays and disruptions).

Each of these steps is a hard problem in itself. Therefore, in practice each step is usually considered in isolation, building on the results of the previous steps and creating input for the next steps. In the last decades, a lot of mathematical optimisation models have been developed for these problems and a variety of optimisation techniques has been applied. However, the increasing demands from practice as well as the availability of cheap and more powerful computation hardware asks for the solution of larger and larger problem instances.

This work has been motivated by one of the planning steps, the creation of operational timetables for passenger and freight trains, the so called *Train Timetabling Problem* (TTP), see, *e. g.*, [15, 71]. In the two BMBF-projects OVERSYS[1] and KOSMOS[2] we worked together with our industrial partner *Deutsche Bahn* on the creation of operational timetables for the German railway network. The goal was to develop techniques for the creation of timetables for a time period of about six hours for all freight and all passenger trains in the German railway network at once. We should not conceal that this was maybe too ambitious a goal: The research in the past showed that the TTP is really challenging. Only in the last few years people started to work on problems that involve not only single corridors but structured railway networks. Furthermore the number of trains is usually much smaller than required for the German network. The following table, created by Schlechte [86], gives a small overview over the considered problem sizes.

[1] BMBF grant 03HEPAG4
[2] BMBF grant 05M10OCD

reference	#stations	#tracks	#trains
Szpigel (1973) [91]	6	5	10
Brännlund et al. (1998) [11]	17	16	26
Caprara et al. (2002) [19]	17	16	221
	102	101	41
Cacchiani et al. (2008) [13]	17	16	221
	102	101	41
Cacchiani et al. (2010) [14]	65	64	775
Fischer and Helmberg (2010) [34]	*1776*	*3852*	*3388*

The largest problem instances that we considered so far are shown in the last row of the table. These instances comprise the south-western subnetwork of *Deutsche Bahn*, roughly Baden-Wuerttemberg, which is about 10% of the whole German railway network. But even these instances are much larger than others considered in literature so far. It is therefore not surprising, that the current techniques and tools are not sufficient to tackle the full German network efficiently.

The goal of this thesis is to develop techniques that help to deal with very large scale problems as they arise in train timetabling but can also be applied to other problems that use similar solution approaches. Indeed, our solution approach follows the most successful ones in the literature for TTP: The schedule of each single train is modelled as a shortest path problem in its associated time expanded network (*e. g.*, [19]). These models lead to Integer Programming formulations of large scale, which are usually far too big to be solved with state-of-the-art solvers for Integer Programming. So the standard approach is to use techniques based on Lagrangian relaxation [11, 21, 64] or Dantzig-Wolfe decomposition [13, 24], in order to get bounds as well as to guide heuristics to solve the problem approximately. We will use Lagrangian relaxation as our main solution approach.

Lagrangian relaxation leads to a convex, non-smooth optimisation problem. Problems of this kind are typically solved by first order methods, which are iterative algorithms. Applied to the Lagrangian relaxation of our problem, they require in each iteration the solution of one shortest path problem in each time expanded network w. r. t. a changing objective function. This changing objective function consists of the original objective and the penalising term induced by the current *Lagrange multipliers*.

While this solution approach is quite promising, we realised early in our project that it is not efficient enough for our application. In particular, two properties of the solution approach caused difficulties.

1. Time expanded networks grow quickly if the number of time steps increases. In consequence, the number of variables and constraints in the IP formulation as well as the networks to be considered in the shortest path subproblems in the Lagrangian relaxation approach are huge.

2. Because of the large number of trains the number of subproblems is large as well (each train leads to one shortest path problem in its associated time expanded network). This means that the total number of subproblem evaluations is also large and the main source of computational burden. However, in a structured network

with different types of trains, not all trains are equally important or involved in the same number of conflicts. Thus, for some trains this large number of evaluations may indeed by required in order to resolve all conflicts, but for other trains a significantly smaller number of evaluations may be sufficient.

In this thesis, we focus on the improvement of the Lagrangian relaxation solution approach. Our main contribution is the development of two techniques that focus on the two points above.

1. We develop a *Dynamic Graph Generation* technique, which allows to construct a shortest path oracle that mimics the behaviour of the full graph on the outside but works with a small subgraph internally that is updated dynamically as needed. Indeed, this technique allows to deal with an infinite number of discretisation steps, which further eases the application of time expanded models. All this works without sacrificing accuracy of the solution approach.

2. Our second contribution is an *Asynchronous Parallel Bundle Method* for subspace optimisation. Instead of solving each subproblem once in each iteration, the algorithm selects only small subspaces of Lagrange multipliers and optimises over the subproblems that are influenced by them. Because such a subspace typically interacts only with a small number of subproblems, several subspaces can be selected in parallel and optimised asynchronously and independently on the associated subsets of subproblems. This leads to a significant reduction of the number of subproblem evaluations.

Next we give an overview over this thesis. In particular, we will present the basic ideas of both of our techniques and sketch our main results. In the second chapter we will formalise our problem and present the models we use. The third chapter discusses general solution approaches for the problems of that type. The main part are Chapters 4 and 5, which discuss the two new techniques that we developed.

Chapter 2: The Train Timetabling Problem

There are many varieties of train timetabling problems in the literature. The most influencing difference is whether one considers periodic [68] or aperiodic [19] problems. The former is the most common type in passenger transport problems, in particular in urban traffic. In contrast, the problem we consider comprises the planning of passenger and freight trains for a railway network of the size of Germany's. Especially freight trains but also long distance passenger trains are largely aperiodic.

In this chapter we give the exact description of the problem we consider. We present the basic modelling approach, which is based on time expanded networks with coupling constraints. Each time expanded network models the schedule for one train along its route w. r. t. running times and stopping times at certain stations. The coupling constraints restrict the simultaneous usage of shared resources, namely stations and tracks. For stations they correspond to the maximal number of trains that can visit a certain station

at the same time. For tracks they model minimal safety distances between two trains running in succession on a common railway track. These distances are often called *headway times*.

The headway time constraints turned out to be the most difficult ones. In the literature there are two approaches for modelling them. The first one is based on inequality constraints that prohibit the simultaneous usage of conflicting arcs in the time expanded networks, see, *e. g.*, [19]. The second approach models feasible selections of train runs over one physical track that are not in conflict, so called *configurations*, using additional configuration networks [9]. In our project we used both approaches and it turned out that both have advantages and disadvantages.

Chapter 3: Solution methods

In this chapter we review the two basic solution approaches for the type of models that arise in our application, column generation [28] and Lagrangian relaxation [64]. Both approaches lead to the same type of subproblems, thus our Dynamic Graph Generation approach can be applied in both (see below).

Our solution approach is based on Lagrangian relaxation and works as follows. The IP formulation consists of two main ingredients. First, the schedule of a single train is modelled as a path in its associated time expanded network (in the second modelling approach, configuration networks are another type of time expanded networks, but the basic structure of the model is the same). The trains compete for common resources like station and track capacities. These restrictions lead to linear *coupling constraints*. Lagrangian relaxation then relaxes these coupling constraints by considering the problem where, instead of enforcing these constraints, their violation is penalised in the objective function. This penalty depends on the so called *Lagrange multipliers*. The algorithmic approach is then to find optimal (or nearly optimal) multipliers that provide the "best" possible penalisation giving the best Lagrangian bound for the problem. Determining optimal multipliers is a non-smooth convex optimisation problem. These kinds of problems are typically solved by so called first order methods, in our case by a proximal bundle method [6].

Chapter 4: Dynamic Graph Generation

One of the major difficulties is the enormous size of the time expanded networks. When dealing with such models, one usually has to find a compromise between the number of time steps to be contained in the model and the number of variables and constraints: the number of variables (which correspond to the arcs in the networks) grows quickly if the number of time steps is increased. This has been a problem for the large networks in our models, too, which contain thousands of trains and stations and several hundred time steps (*e. g.*, a horizon of six hours and a discretisation of one minute lead to 360 time steps). In fact, these networks were so large that we could hardly handle them in the computers we used at the beginning of our project. This was even more threatening in

view of the ambitious aim of our project to deal with the entire German network at once, possibly using a finer discretisation than one minute.

Fortunately, the shortest path problems to be solved are not completely unpredictable even if the objective function changes in each iteration. A fundamental objective in our project, and in many timetabling and scheduling problems in general, is that the trains should arrive at their destination as early as possible. A consequence is that most of the trains only use the early time steps and only few trains have longer waiting periods, although it is not clear from the beginning which trains have to wait (it is part of the problem to figure out which trains have to wait, where they have to wait and how long).

Dynamic Graph Generation exploits this observation. The basic objective function encourages trains to run in the early time steps. During the solution process, the changing Lagrange multipliers induce augmented cost terms if the current solution violates some constraints so that this violation is reduced in the next iteration. This means that the objective function in areas of conflicts may change arbitrarily, but the underlying objective, the preference of early runs without delays, still dominates on most parts of the networks. Consequently, the shortest path w. r. t. the current Lagrange multipliers is usually also in the early time steps of the network. The Dynamic Graph Generation technique to be presented stores only the early time steps of the network. Using a specially adapted objective function for this smaller network, we show that either the problem on the small subnetwork solves the shortest path problem on the whole network, too, or we get the information that the network is too small and must be expanded. In the latter case the algorithm dynamically generates new parts of the network to be stored in memory during the solution process until it can certify that the subproblem is solved correctly.

We will show that the Dynamic Graph Generation approach can be used to solve the shortest path problem in time expanded networks by only storing a small part of them. This part mainly consists of all shortest paths being generated in the earlier iterations (of course, these paths must be contained in the stored subnetwork) plus a bounded number of additional time steps. We will prove that this bound is rather small and depends only on the running times of trains (which are a fixed part of the input data) in common cases. The Dynamic Graph Generation algorithm itself, which determines in each iteration which parts of the network must be added dynamically, is very efficient and its running time only depends linearly on the length of the train route and the running times of the trains but not on the size of the subnetwork being stored (in contrast to the shortest path computation).

Because we only store a small part of the subnetworks, it is possible to work with models that contain an infinite number of time steps. This is very convenient in practical applications because one does not need to determine a maximal time horizon a priori. Instead, the subnetworks grow dynamically to the required size. In general, dealing with infinite networks may endanger the finiteness of some algorithms. We show that under reasonable assumptions the solution process of the subproblem is finite. Even more, together with the boundedness of the stored subnetwork, this expansion is not much larger than the subnetwork that contains all shortest paths computed in some iteration.

An important property of our approach is that the shortest path computations are exact. This means, although only a small part of the time expanded network is kept in

memory and taken into account by the shortest path algorithm, the computed shortest path is always guaranteed to be a solution of the complete subproblem on the whole time expanded network. In consequence, the Dynamic Graph Generation technique can be used as a drop-in-replacement for the traditional subproblem computations working on the whole network. In fact, given the efficiency of our technique, one should always employ Dynamic Graph Generation whenever one deals with shortest paths subproblems in time expanded networks: one gets an improved memory usage and performance of the shortest path algorithm (since it only has to consider small parts of the whole network) without sacrificing any accuracy of the model.

The Dynamic Graph Generation approach has been motivated by our train timetabling application. Nevertheless, it is not restricted to this particular application. It can be applied in any application with time expanded networks and an objective that prefers early paths. In fact, our approach covers more than we require for our projects. In our TTP we assume that the route of each train is fixed and that a train has at most two options at each station: pass through the station without stopping or stop at the station and wait. Dynamic Graph Generation is more general in both aspects. First, it can be applied if, in addition to the scheduling decisions, certain routing decisions can be made. In the case of TTP these may be potential alternative routes. Second, there may be more than two choices at a station. For example, a train could use different tracks within a station, some tracks requiring more time. Another example would be the planning of a manufacturing process, where each station corresponds to one production step and each step can be done with different tools or at different working temperatures.

Chapter 5: Asynchronous Parallel Bundle Method

The Lagrangian relaxation approach leads to a convex, non-smooth optimisation problem, where the objective function $f\colon \mathbb{R}^M \to \mathbb{R}$ is given by a *first order oracle*. This means that, given a certain point $y \in \mathbb{R}^M$, the oracle determines the function value $f(y)$ and a subgradient $g \in \partial f(y)$. Non-convex optimisation problems given by a first order oracle are typically solved by first order methods, like a subgradient or bundle method, to determine an optimal or nearly optimal solution y^*.

If the convex optimisation problem arises from Lagrangian relaxation, the main computational burden is the evaluation of f at certain points y. With R denoting the set of all trains, the objective function f has the form

$$f(y) = \sum_{r \in R} f_r(y),$$

where each function f_r, $r \in R$, is itself a non-smooth convex function. The evaluation of f_r requires then, in our case, the solution of one shortest path problem in the corresponding time expanded network. Therefore, each single evaluation, *i. e.*, each call to the oracle, requires the solution of one shortest path problem in each network.

The goal of the asynchronous parallel bundle method presented in this work is to reduce the number of evaluations of the subproblems and therefore to improve the performance

of the overall optimisation process. The idea of the algorithm is to exploit the structural dependencies between the subproblems, to be sketched in the following. In Lagrangian relaxation (see above), the variables y are the Lagrange multipliers of the coupling constraints. A subproblem $r \in R$ (*i. e.*, the function f_r) depends only on a certain y_j, $j \in M$, if the corresponding coupling constraint interacts with the subproblem r. For example, in the TTP a station capacity constraint only interacts with trains that actually visit this station, whereas all other trains are not influenced by this constraint. Indeed, in the large networks that we consider most trains do not share a common constraint at all (think of regional trains running in the north or the south). In other words, each function f_r, $r \in R$, depends only on a small subset of constraints resp. their Lagrange multipliers.

The idea of our method is to select small subspaces of Lagrange multipliers that promise a large progress. Such a subspace corresponds to a subset of coupling constraints that are strongly violated in the current point (*i. e.*, w. r. t. the current Lagrange multipliers). The algorithm then collects these constraints together with the subproblems (trains) that interact with them and forms a *subspace problem*. The subspace problem only optimises over the chosen variables and keeps all other variables fixed. Then a subprocess uses a standard proximal bundle method to solve the subspace problem, *i. e.*, it tries to reduce the violation of the chosen constraints. If the set of constraints is small and these constraints couple only few subproblems, then the subspace problem only works on few constraints and trains. In this case the algorithm may select several disjoint subspaces and optimise over each of them independently and in parallel. Certain subspaces chosen this way can be larger or contain more difficult subproblems than others. The parallel bundle method does not wait until the optimisation on some subspace is completed. Instead, it keeps selecting further subspaces and starts new processes solving them. Whenever a certain process is done, its solution is incorporated in a global solution, *e. g.*, the solution associated with its subspace is written to the global data. Because the algorithm does not wait until some process finishes, it behaves in a completely asynchronous manner.

Optimising over a subspace and ignoring the rest can endanger convergence. If a process resolves some conflicts between a few trains locally, these trains will be assigned new schedules. Although these new schedules will have smaller conflicts at the chosen constraints, they may introduce new conflicts with other constraints, possibly with other trains that have not been considered by the process. The parallel bundle algorithm automatically detects dependencies that may endanger convergence during the solution process. If such a dependency is detected, further subspace selections will respect these dependencies.

We will show that the asynchronous parallel bundle method with automatic dependency detection leads to a convergent algorithm. We will prove the convergence of both, the Lagrange multipliers and a convex combination of primal subproblem solutions to optimal solutions of the dual resp. primal convex relaxation.

The basic method relies on the property, that each single coupling constraint interacts only with few subproblems. This is not true in general. For example, a station capacity constraint at a certain time index couples *all* trains using this station, although trains that run early in the morning will hardly have a conflict with trains that run in the evening. In order to make our method more suitable for practical situations, we developed

an extended version of our parallel bundle method. In addition to the dependency detection between coupling constraints, this algorithm also detects automatically which subproblems do actually interact with which constraints during the solution process (*i. e.*, which restrictions are important for which train). These dependencies will again influence future subspace selections and the creation of subspace problems. Analogous to the basic method we will prove convergence of dual and primal solutions to respective optimal solutions.

The asynchronous parallel bundle method is also motivated by the TTP and the observation that many trains have only local conflicts. But like Dynamic Graph Generation, this approach can be used for more general applications. In fact, any solution approach that involves a Lagrangian relaxation to be optimised by subgradient methods can also use the parallel bundle method. In contrast to Dynamic Graph Generation, the subproblems do not even have to be shortest path problems. In the worst case, if the couplings between the subproblems are too strong, our method will behave like a classical proximal bundle method. However, if the problem consists of a large number of loosely coupled subproblems, then it can give a significant improvement in terms of required subproblem evaluations and therefore improve convergence.

Chapter 6: Numerical tests

At the end of this thesis we present some short numerical tests to illustrate the developed techniques. The purpose of these tests is not to provide a computational study of the TTP and is also not an in depth numerical study of the developed algorithms. Instead, the main focus of this thesis is the development of theoretic foundations for algorithmic approaches that are motivated by the requirements for solving very large scale problems like the TTP. The numerical tests should be considered as a demonstration of our techniques and illustrate their potentials in practical large scale applications.

We tested our Dynamic Graph Generation technique at real world instances of *Deutsche Bahn*. In these tests we achieved a reduction of the number of arcs (corresponding to integer variables) in the time expanded networks by a factor of about 40.

The parallel bundle method is demonstrated on a set of randomly generated TTP instances of large size. The numerical tests show that our preliminary implementation (from a technical, computer science point of view) reduces the number of subproblem evaluations together with the running times significantly.

We also present some test results for our largest real world instances. Our Lagrangian relaxation approach can be used to solve these instances in half a day with less than 1% of all passenger trains suffering a significant delay and a large reduction of the arrival times of the freight trains of one hour on average.

Notation

In this section we introduce some notation that we will use throughout the thesis. The sets $\mathbb{N}$, $\mathbb{N}_0$, $\mathbb{Z}$ and $\mathbb{R}$ refer to the natural numbers, the natural numbers with zero, the integers and the real numbers and a subscript $+$ $(-)$ refers to the corresponding nonnegative (nonpositive) subset. Let M be a discrete set, then $\mathbb{R}^M$ denotes the real vector space of all vectors with components in M. As usual, the component of $x \in \mathbb{R}^M$ corresponding to some $j \in M$ is denoted by x_j. We extend this notation to sets of indexes: given a subset $J \subseteq M$, the subvector of the components of x belonging to J is $x_J = (x_j)_{j \in J}$. Let R be a finite set. We will often deal with matrices of the form

$$C = [C_{\bullet,r}]_{r \in R} = \left[C_{\bullet,r_1}, \ldots, C_{\bullet,r_{|R|}}\right] \in \mathbb{R}^{M \times n}$$

where $n = \sum_{r \in R} n_r$. Similarly to vectors, given subsets $J \subseteq M$ and $R' \subseteq R$ we refer to the lines J of blocks R' by $C_{J,R'}$. If a set contains only one element, *i. e.*, $J = \{j\}$ or $R' = \{r\}$ then $C_{j,r}$ denotes the j-th row of block r.

Let $x, c \in \mathbb{R}^M$ be two vectors, then we denote the inner product of x and c by

$$\langle c, x \rangle = c^T x = \sum_{j \in M} c_j x_j,$$

and the Euclidean norm is

$$\|x\| = \sqrt{\langle x, x \rangle}.$$

Let $X \subset \mathbb{R}^M$ be a set. The linear hull (or the span) of X is denoted by

$$\operatorname{span} X := \left\{ \sum_{i=1}^{m} \lambda_i x_i \colon \lambda_i \in \mathbb{R}, x_i \in X, i \in \{1, \ldots, m\}, m \in \mathbb{N} \right\},$$

and the convex hull of X is denoted by

$$\operatorname{conv} X := \left\{ \sum_{i=1}^{m} \lambda_i x_i \colon \sum_{i=1}^{m} \lambda_i = 1, \lambda_i \geq 0, x_i \in X, i \in \{1, \ldots, m\}, m \in \mathbb{N} \right\}.$$

For an arbitrary function $f \colon Y \to \mathbb{R}$ and a set $Y' \subset Y$ we denote by $\operatorname{Argmin}\{f(y) \colon y \in Y'\}$ the (possibly empty) set of all minimisers of f over Y', and similarly by $\operatorname{Argmax}\{f(y) \colon y \in Y'\}$ the (possibly empty) set of all maximisers. If these sets contain exactly one element we write argmin resp. argmax.

The subproblems we deal with are shortest path problems in time expanded networks. We will use the following notation from graph theory, see, *e. g.*, Diestel [29] for an introduction. A *directed graph* $G = (V, A)$ is a pair of a set of nodes V and a set of arcs $A \subseteq V \times V$. Note that sometimes we allow loops $(v, v) \in A$ for some $v \in V$. We will often write $uv \in A$ for an arc $(u, v) \in A$. The set of nodes of a graph G is denoted by $V(G)$, the set of arcs by $A(G)$. A graph $G' = (V', A')$ is called a *subgraph* of G, write $G' \subseteq G$, if $V' \subseteq V$ and $A' \subseteq A$. It is an *induced subgraph* if $u, v \in V' \wedge uv \in A \Rightarrow uv \in A'$. Let

$V(G)$, $A(G)$

$G[V']$ $V' \subset V$ be a subset of nodes. Then $G[V']$ denotes the subgraph of G induced by V', *i. e.*, $V(G[V']) = V'$ and $A(G[V']) = \{(u, v) \in V' \times V' \colon (u, v) \in A(G)\}$.

A *walk* $P = u_1 \dots u_m$ in G is a sequence of nodes $u_1, \dots, u_m$, with $u_i u_{i+1} \in A$ for all $i \in \{1, \dots, m-1\}$. If the nodes are pairwise disjoint, P is called a *path*. The set of nodes of P is denoted by $V(P) = \{u_1, \dots, u_m\}$, the set of arcs by $A(P) = \{u_i u_{i+1} \colon i \in \{1, \dots, m-1\}\}$. If there is no danger of confusion we will simply write $u \in P$ or $uv \in P$ for a node u resp. an arc uv contained in P. The subwalk (resp. subpath) of walk (path) P starting at u_i and ending in u_j, $1 \leq i \leq j \leq m$ is denoted by $u_i P u_j$ (note that we explicitly allow $i = j$, in this case $u_i P u_j$ is a path of only one node). If $i = 1$ or $j = m$ we will write $P u_j$ respectively $u_i P$ for the subwalk. For two walks $P = u_1 \dots u_m$ and $Q = v_1 \dots v_{m'}$ with $u_m = v_1$ we denote by $PQ = P u_m Q = P v_1 Q$ the walk constructed by first following P and then continuing using Q.

Acknowledgement

I would like to thank all people who supported me during the last years. Special thanks go to my supervisor, Prof. Dr. Christoph Helmberg, for his guidance and all the mathematical disputes that helped me a lot during my research. Furthermore I wish to acknowledge the financial support by the *Bundesministerium für Bildung und Forschung* and the practical partner *Deutsche Bahn* in our project, which gave me the opportunity to work on the interesting train planning problems that motivated my work. I thank all my colleagues for creating a very kind atmosphere and giving me insides in other areas as well.

Finally I want to thank my family and especially my wife Anja for her love and for uplifting me when I was down and believing in me all the time, and our cats, who reminded me that there is always some time to relax and to enjoy life.

2 The Train Timetabling Problem

2.1 Introduction

The main motivating problem for this work is the train timetabling problem TTP. This problem can be stated as follows. We are given a railway infrastructure network and a set of trains with predefined routes in the network. The network possesses several restrictions like station capacities or safety distances between successive trains on the same track (headway times). Each train has furthermore a certain speed that allows to travel over each track in a certain amount of time. The aim is to find a conflict-free schedule of all trains in this network that observes all restrictions. Besides the basic restrictions mentioned above (stations and track capacities) there may be further restrictions depending on the concrete instances. Typical restrictions are

- periodicity of the schedules (especially for passenger trains),
- small deviation from a predefined schedule (delay management),
- time windows in which certain trains may use some track,
- connectivity restrictions between (passenger) trains at certain stations (in order to allow passengers to change trains),
- ...

The most important and most influential property is the first one, periodicity. Particularly in passenger train timetabling (in cities this holds for subway, bus, tram, ...) the aim is a periodic timetable where trains repeat their schedule, *e. g.*, each hour. This kind of problem requires different models and techniques. The topic of this work focuses on *aperiodic* timetabling, especially because of the presence of non-periodic freight trains.

Planning problems for railway networks have gained a lot of interest in the literature in the last decades. The planning process usually consists of several steps like Network Planning, Line Planning, Timetable Generation, Track Allocation, Rolling Stock Scheduling, Crew Scheduling and Real Time Decision Management. The timetabling problem, which we consider, lies between the Line Planning step, which determines the routes for each train, and the Track Allocation step, which assigns concrete physical tracks to trains, *e. g.* in stations. Thus, we consider the lines as being given and do not deal with the concrete track allocation in stations (we consider a station as a single node in the infrastructure network, although a large station could be represented by several nodes). The focus of train timetabling is on the generation of feasible schedules of all trains. An overview over the different planning steps can be found in Caprara et al. [20], recent surveys focused on

the timetabling and track allocations steps are Lusby et al. [71] and Cacchiani and Toth [15]. We will give a short overview over the literature focused on timetabling.

Models for periodic timetable problems are usually based on the *Periodic Event Scheduling Problem* (PESP) which has been introduced by Serafini and Ukovich [89]. These models assign certain events continuous time variables that represent the point in time of this event in a periodic horizon, *i. e.*, the event takes place at times $t, t + T, t + 2T, \ldots$, where T denotes the length of the period. A remarkable property of the cyclic nature of these problems is that there is no real order of each two events: each event appears before and after each other event w. r. t. the periodicity. Thus, the most important constraints in the PESP models restrict the (cyclic) distance between two events i and j, *i. e.* a constraint c is of the form

$$l_c \leq \pi_i - \pi_j + z_c \cdot T \leq u_c$$

where l, u are lower resp. upper bounds on the difference between those two events. The integer variable $z \in \mathbb{Z}$ models the periodicity of the events. In the case of train timetabling the events correspond to arrivals and departures of the trains at some station, the constraints model, *e. g.*, minimal distances (headway times) between the departure events of successive trains. Peeters [80] showed how station capacities, which cannot be expressed directly by bound constraints above, can be handled in PESP models as well. The PESP has been successfully used in passenger traffic planning, see Liebchen [68], Liebchen and Möhring [69] for the Berlin underground, or Kroon et al. [62] for the Dutch railway.

In contrast to passenger traffic, which is usually periodic, freight traffic is often non-periodic. If the desired timetable has an underlying periodic structure, then PESP models can be used to reserve certain periodic corridors, which can be assigned to freight trains afterwards. This approach is investigated by Caimi [16], Caimi et al. [17, 18] for Switzerland.

In large scale networks with a lot of freight trains the timetable has often a quite non-periodic nature. There are typically two types of models for this kind of problems. The first one assigns each arrival and departure event a continuous variable modelling the time of this event, similar to PESP models. Because of the lack of periodicity, each two events have an explicit order in which they occur. This order, which event comes first, is modelled by binary decision variables. These decision variables are then coupled with the continuous time variables using big-M constraints, see, *e. g.*, Carey and Lockwood [22], Higgins et al. [54]. The downside of big-M approaches is that the bound obtained by the corresponding LP relaxation is often quite weak.

The second type of models is based on discretisation of the time horizon (into, say, minutes). These models are based on graph formulations, where each node corresponds to an event at a specific (discrete) point in time. Two nodes, belonging to the same train, are connected by an arc if the two corresponding events may appear in order, *e. g.*, the arrival and departure at some station with an appropriate delay for waiting. A schedule of a train is then represented by a path in this network, the arrival and departure events of the train correspond to the nodes on the path. This leads to a multi-commodity flow formulation in which each arc is assigned a binary decision variable denoting whether

the arc is contained in the path. Brännlund et al. [11] present a model of this kind for a train timetabling problem in a single line and applied a Lagrangian relaxation approach in order to find good meet/pass decisions for the trains so that unnecessary waiting time is minimised. Caprara et al. [19] use a similar approach, but add further variables that correspond to trains visiting certain nodes (instead of arcs) and formulate constraints like overtaking restrictions in terms of these variables. The constraints are equivalent to so called packing constraints that forbid the simultaneous use of conflicting arcs. The aim is to find a feasible timetable that is close to a given "ideal timetable" for each train. The approach is extended in Caprara et al. [21] to model further constraints that are important in practice, *e. g.*, station capacities, prescribed timetables and maintenance operations. Cacchiani et al. [13] presented a column generation approach based on a path formulation of a variant of the model of Caprara et al. [19, 21], in which the binary decision variables are assigned to each possible path (*i. e.*, each possible schedule) of a train. Due to the large number of variables column generation has to be employed. They could show that the path based formulations lead to stronger (linear) relaxation than the arc based formulations. In [14] a variant of this model is applied to the case where additional freight trains must be integrated in an existing timetable, consisting usually of passenger trains. While the timetable for the passenger trains is fixed, the freight trains are given an ideal timetable and the task is to find a feasible schedule for these trains that is as close as possible to that timetable.

Borndörfer et al. [7], Borndörfer and Schlechte [9] consider a timetabling problem not only on a corridor but on a structured infrastructure network. They present different models based on arc and path formulations. Furthermore, in Borndörfer and Schlechte [9], the authors introduce an alternative formulation for "blocking" or "headway" constraints in terms of "configurations". A configuration is a feasible set of arcs corresponding to conflict free runs on a certain infrastructure arc. Instead of forbidding the simultaneous use of conflicting arcs by packing constraints, a configuration explicitly allows only feasible choices of arcs. Borndörfer and Schlechte show that configuration based relaxations lead to better bounds than packing formulations.

The problem that we consider is highly non-periodic. Because of this we follow the modelling approach of Borndörfer and Schlechte [9], Caprara et al. [19] using graph based, time discretised formulations. Like Borndörfer and Schlechte [9] we will consider structured networks with single line tracks and additional constraints. We will also take into account a special form of a prescribed timetable for passenger trains. However, unlike Cacchiani et al. [14], the timetables of passenger trains are not fixed. We merely assume that each passenger train is assigned a certain time window that specifies when and how long a train should wait at a certain station. Note that this is not the same as a completely given ideal timetable. In fact, these windows are only given at stopping stations (especially long-distance trains may pass certain stations without stopping), and there is no guarantee that there is a feasible timetable for a single train that satisfies all of its time windows. In our project we considered models based on packing constraints as well as configuration based models like Borndörfer and Schlechte [9]. In contrast to most approaches in the literature, we assume that headway times and running times of a train depend on the type of the train as well as on its stopping behaviour, *i. e.*, on whether a

train stops at a station or passes through the station. While the stopping behaviour is not important for passenger trains, which usually have a fixed set of stations at which they must stop, it makes a huge difference for freight trains. For freight trains the decision where they have to stop and let faster trains pass is crucial. It turned out that these behaviour dependent headway and running times lead to much more complicated packing constraints resp. configuration networks.

This chapter is organised as follows. In Section 2.2 we give an abstract description of the timetabling problem we consider. In particular, we discuss the unique requirements like behaviour and type dependent running and headway times as well as the prescribed timetables for passenger trains. Afterwards in Section 2.3 we present the basic model based on time expanded networks. This leads to a basic formulation as Integer Program in Section 2.4. In our project we considered two modelling approaches from the literature. The first, based on packing constraints, is presented in Section 2.5, the second configuration-based formulation is given in Section 2.6. Finally in Section 2.7 we discuss the objective function that we used in our application.

The problem description and models presented in this chapter are based on the articles [36] and [34]. The presentation has been extended and unified, the objective function is defined in a different way.

2.2 Problem Description

In this section we describe the concrete TTP problem that we worked on in the BMBF-projects OVERSYS and KOSMOS. In both projects we worked together with our industrial partner *Deutsche Bahn AG*.

The overall problem can be stated as follows. We are given an infrastructure network $G^I = (V^I, A^I)$ with set of nodes V^I and set of directed arcs A^I:

$G^I = (V^I, A^I)$

V^I

- $V^I, |V^I| \leq \infty$ ... stations, track switches, ...,

A^I

- $A^I \subseteq V^I \times V^I$ ... tracks.

We assume that each physical track is represented by one directed arc for each direction in which it may be used. This means that there are two arcs $(u, v), (v, u) \in A^I$ for a typical physical track connecting u and v. The set of infrastructure arcs is partitioned into *single track* arcs and *double track* arcs

- $A^I = A^{I,1} \dot\cup A^{I,2}$ with

$A^{I,1}$

  - $A^{I,1}$ ... single tracks,

$A^{I,2}$

  - $A^{I,2}$ ... double tracks.

  We denote the arc in the opposite direction of $a \in A^I$ with $a^{-1} \in A^I$. For some $a = (u, v) \in A^I$ with $a^{-1} = (v, u) \in A^I$ we have $a \in A^{I,1} \iff a^{-1} \in A^{I,1}$. In other words, if one arc is a single track, then the reverse arc (if exists) is also a single track.

Single tracks correspond to only *one* physical track. This means that trains coming from both directions use the same physical track. Therefore, if one train runs over the arc, say, $a \in A^{I,1}$, then no other train may use the arc $a^{-1} \in A^{I,1}$ at the same time because both arcs correspond to the same physical track. In contrast, double tracks have one physical track for each direction. Therefore when one train uses the track in some direction $a \in A^{I,2}$, another train may use the track in the opposite direction $a^{-1} \in A^{I,2}$ simultaneously.

Each node is assigned an *absolute capacity* $c_u \in \mathbb{N}$, $u \in V^I$. This is the maximal number of trains that are allowed to be at u at the same time. One can think of the absolute capacity as the number of different tracks in stations. Similarly we assign each arc a *directional capacity* $c_a \in \mathbb{N}$, $a = (u, v) \in A^I$. The directional capacity is the maximal number of trains that are allowed to be at v at the same time *arriving from* u. In some stations a certain track may be reserved for trains arriving from a specific direction. Therefore the directional capacity can be used to model restrictions of this kind. In fact, the absolute capacity and the directional capacity together suffice to model the track layout of small stations (with only two possible directions). Table 2.1 shows some different example layouts of small stations along with their respective absolute and directional capacities.

absolute capacity

directional capacity

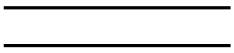
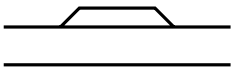
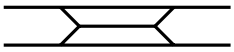
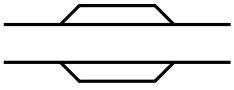
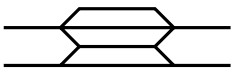
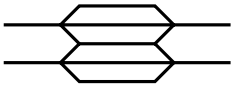
Layout	absolute capacity	left directional capacity	right directional capacity
	**2**	**1**	**1**
	**3**	**1**	**2**
	**3**	**2**	**2**
	**4**	**2**	**2**
	**4**	**2**	**3**
	**5**	**3**	**3**

Table 2.1: Absolute and directional capacities for small stations with two possible directions. In this work we assume that trains run always on the right-hand track in their driving direction (this is true for Germany and large parts of Europe).

In our problem we distinguish several train types. Each train belongs to exactly one train type. The most important distinction are *passenger* trains and *freight* trains.

Furthermore the passenger trains are partitioned in *short distance* and *long distance* passenger trains. Each train is assigned a *route*, *i. e.* a unique sequence of nodes in the infrastructure network. Formally we have

Θ
- $\Theta = \Theta^p \dot\cup \Theta^f$ ... set of train types with
  - Θ^p $\Theta^p = \Theta^{\mathrm{ld}} \dot\cup \Theta^{\mathrm{sd}}$ ... set of *passenger* train types, with
    - Θ^{ld} Θ^{ld} ... set of *long distance* train types,
    - Θ^{sd} Θ^{sd} ... set of *short distance* train types,
  - Θ^f Θ^f ... set of *freight* train types,

R
- R ... a set of *trains*,
- $\theta\colon R \to \Theta$ ... $\theta(r)$ train type of train $r \in R$,

$\bar G_r = (\bar V_r, \bar A_r)$
- $\bar G_r = (\bar V_r, \bar A_r) \subseteq G^I$ ... *route* of $r \in R$ where
  - $\bar V_r$ $\bar V_r = (u^r_1, \dots, u^r_{|\bar V_r|})$ ... nodes that r has to visit,
  - $\bar A_r$ $\bar A_r = \{(u^r_i, u^r_{i+1})\colon i \in \{1, \dots, |\bar V_r| - 1\}, u^r_i, u^r_{i+1} \in \bar V_r\}$ ... the tracks that r must use.

  In other words, $\bar G_r$ is a walk in the infrastructure network.

The *running time* of a train over a track may differ significantly depending on whether the train stops at the start or end node of the track or if it passes through the nodes without stopping. In order to model these differences we define the set of possible *stopping behaviours* and *running behaviours*,

B^{STOP}
- $B^{\mathrm{STOP}} := \{Run, Stop\}$,

B
- $B := B^{\mathrm{STOP}} \times B^{\mathrm{STOP}} \simeq \{RunRun, RunStop, StopRun, StopStop\}$.

Similarly the *headway times* between two trains on a track depend, in addition to their respective train types, on the running behaviours of both trains. Formally running times and headway times are given as functions

t^R_a
- $t^R_a\colon \Theta \times B \to \mathbb{R}_+$, $a \in A^I$, ... $t^R_a(\theta, b)$ is the *running time* of a train with type θ and running behaviour b over arc a,

t^{HW}_a
- $t^{HW}_a\colon (\Theta \times B) \times (\Theta \times B) \to \mathbb{R}_+$, $a \in A^I$, ... $t^{HW}_a(\theta_1, b_1, \theta_2, b_2)$ is the *headway time* over arc a when a train of type θ_1 with running behaviour b_1 precedes a train of type θ_2 with running behaviour b_2,

$t^{HW^{-1}}_a$
- $t^{HW^{-1}}_a\colon (\Theta \times B) \times (\Theta \times B) \to \mathbb{R}_+$, $a \in A^{I,1}$, ... $t^{HW}_a(\theta_1, b_1, \theta_2, b_2)$ is the *headway time* over *single track* $a = (u, v)$ when a train of type θ_1 with running behaviour b_1 running in direction of a precedes a train of type θ_2 with running behaviour b_2 running in direction of $a^{-1} = (v, u)$.

We will assume throughout this work that running times and headway times fulfil certain regularity conditions, which are quite reasonable from a practical point of view. First, the running time of a train increases if it stops at one or both adjacent nodes. We call this monotonicity of running times. monotonicity

$$\begin{aligned}\forall\, a \in A^I, \theta \in \Theta\colon t_a^R(\theta, RunRun) &\leq \min\{t_a^R(\theta, StopRun), t_a^R(\theta, RunStop)\}\\ &\leq \max\{t_a^R(\theta, StopRun), t_a^R(\theta, RunStop)\}\\ &\leq t_a^R(\theta, StopStop).\end{aligned}$$

The second assumption is that the headway times satisfy the triangle inequality. Here this means that the headway time of train r_1 running before r_3 is not larger than the sum of the headway times of r_1 running before r_2 and r_2 running before r_3. This assumption typically holds for technical headway times. We state the condition for *all* possible combinations of three trains, running in the same direction or in opposite directions on a single track. triangle inequality

For all $a \in A^I$, all $\theta_i \in \Theta$ and all $b_i \in B$, $i \in \{1, 2, 3\}$, it holds

$$t_a^{HW}(r_1, b_1, r_3, b_3) \leq t_a^{HW}(r_1, b_1, r_2, b_2) + t_a^{HW}(r_2, b_2, r_3, b_3).$$

In the case of single tracks we have in addition the following triangle inequalities, depending on which of the three trains runs in opposite direction of the other two (we assume that r_1 always uses the track in direction of a),

$$\begin{aligned}t_a^{HW^{-1}}(r_1, b_1, r_3, b_3) &\leq t_a^{HW^{-1}}(r_1, b_1, r_2, b_2) + t_{a^{-1}}^{HW}(r_2, b_2, r_3, b_3),\\ t_a^{HW}(r_1, b_1, r_3, b_3) &\leq t_a^{HW^{-1}}(r_1, b_1, r_2, b_2) + t_{a^{-1}}^{HW^{-1}}(r_2, b_2, r_3, b_3),\\ t_a^{HW^{-1}}(r_1, b_1, r_3, b_3) &\leq t_a^{HW}(r_1, b_1, r_2, b_2) + t_a^{HW^{-1}}(r_2, b_2, r_3, b_3).\end{aligned}$$

We will later see that this assumption is not only convenient and reasonable from a practical point of view but also essential for the validity of some models.

Finding a feasible timetable, *i. e.* a timetable that satisfies all capacity and headway restrictions, is the primary goal of the TTP. In addition there are usually some requirements that prefer certain structures in the generated timetables. In our case we have different requirements for passenger and freight trains.

In passenger train planning there are often certain connectivity conditions between different trains, *e. g.*, two specific trains should be at a certain station at the same time so that passengers can easily switch between those two trains. In order to model such conditions one would require some statistical information about (predicted) passenger flows. For our particular problem there is no such information available. We only have a rough estimate when each passenger train should visit each station. This means that we know for each passenger train and each of its stations a *stopping interval* that specifies *when* the train should stop at the station and a *minimal stopping time* that specifies *how long* the train should stop at the station. Formally, for each passenger train $r \in R$, $\theta(r) \in \Theta^p$, and each station $u \in \bar{V}_r$ we have

$I_r^u = [\underline{I}_r^u, \bar{I}_r^u]$

- a *stopping interval* $I_r^u = [\underline{I}_r^u, \bar{I}_r^u]$ with $\underline{I}_r \in \{-\infty\} \cup \mathbb{R}$ and $\bar{I}_r^u \in \{+\infty\} \cup \mathbb{R}$, $\underline{I}_r^u \leq \bar{I}_r^u$, and

$\delta_r^u \in \mathbb{R}_+$

- a *minimal stopping time* $\delta_r^u \in \mathbb{R}_+$.

The interpretation is the following. If $\delta_r^u > 0$, then train r must stop and wait at the station u at some point within the stopping interval I_r^u and from that point on it must wait for at least δ_r^u seconds. This implies that the train must arrive at the station no later than the end $\bar{I}_r^u$ of the stopping interval and it must not leave earlier than $\underline{I}_r^u + \delta_r^u$. Figure 2.1 illustrates some allowed situations. Note that not all nodes that are visited by the train are necessarily stations. But usually only stations have stopping restrictions. If a node $u \in \bar{V}_r$ does not correspond to a station, then its stopping interval equals $I_r^u = [-\infty, +\infty]$ and the minimal stopping time is $\delta_r^u = 0$. The (first possible) start time I_r^0 of a train at its first station u_1 is specified by a stopping interval $I_r^{u_1} = [\underline{I}_r^{u_1}, \bar{I}_r^{u_1}]$ with $I_r^0 = \underline{I}_r^{u_1} = \bar{I}_r^{u_1}$ and a zero stopping time $\delta_r^{u_1} = 0$.

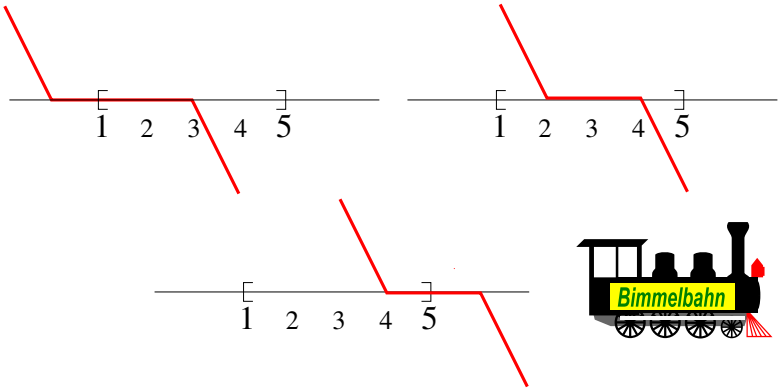


Figure 2.1: Three possible stopping configurations. The train has a stopping interval $I_r^u = [1, 5]$ and a minimal stopping time $\delta_r^u = 2$. It may arrive before the beginning of the stopping interval or at some time within the interval. The train must then wait for the minimal stopping time but must not leave before $\underline{I}_r^u + \delta_r^u$.

For freight trains the situation is different. Freight trains have no stopping conditions at intermediate stations. They only have a (first possible) start time I_r^0 at their respective first station and the objective to reach their final destination as early as possible. We model these (non-)stopping conditions in the same way as for passenger trains:

$$I_r^u = \begin{cases} [I_r^0, I_r^0], & \text{if } u = u_1^r, \\ [-\infty, +\infty], & \text{otherwise,} \end{cases} \qquad \text{and} \qquad \delta_r^u = 0, u \in V_r.$$

Note that in our models we will not *enforce* the observance of the stopping intervals completely. In fact, we will enforce that no train leaves a station too early but we allow that a train arrives too late (before the end of the corresponding stopping interval). In

this case we will penalise the additional delay. The reason is that otherwise it will be very difficult or even impossible to find a feasible solution (it may not exist).

The objective for our TTP is rather vague and, in the end, a modelling detail. The overall policy can be summarised as follows:

1. passenger trains should reach each (stopping) station as early as possible,
2. passenger trains should not wait on non-stopping stations,
3. freight trains should reach their final destination as early as possible (stopping at intermediate stations is allowed and not necessarily penalised),
4. passenger trains should be preferred.

2.3 Model

In this section we describe our model of the TTP, which is similar to the models in [9, 11, 19]. The model is based on *time expanded networks*. The timetable of each single train is modelled by a time expansion of its route graph $\bar{G}_r$ so that timetables correspond to certain paths in that network. The time expanded networks of all trains are then coupled using some linear constraints that enforce the capacity and headway restrictions. We introduce the time expansion of a graph rather generally as it will become useful in later chapters as well. Afterwards we will apply this time expansion to so called base train graphs G_r, $r \in R$, which are strongly related to the route graphs $\bar{G}_r$, $r \in R$.

Time expansion. Let $G = (V, A)$ be a directed graph with loops (*i. e.*, arcs of the kind $uu \in A$, $u \in V$, are allowed) and $T = \{1, 2, \ldots, T^{\max}\}$ the set of discretised time steps with $T^{\max} \in \mathbb{N} \cup \{\infty\}$. Typical choices for the discretisation are one minute, five minutes, ten minutes, 10 seconds, 1 second, and so on depending on the problem at hand. In case of the TTP we will always use a discretisation of one minute, *i. e.*, the time steps of T correspond to the minutes in the planning horizon (with $1 \in T$ being the starting time).

Remark 2.1 Note that we allow T to be an *infinite* set, which is rather unusual for time expanded models. The common approach is to fix $T^{\max}$ to a reasonably large time index so that the important horizon of the planning problem is covered. For the moment it is not clear that we can actually deal with an infinite number of time steps because this leads to models of infinite size. Indeed, we will require *dynamic graph generation* techniques to be developed in Chapter 4 in order to deal with infinitely many time steps. In the present chapter the infinity of the number of time steps is not important for the presentation of the models, just keep in mind that we need the techniques from Chapter 4 to actually solve those models.

The time expansion of G contains one copy of each node of V for each time step of T. These nodes are then connected according to the arcs A and a *traversal-time-function* $d\colon A \to 2^{\mathbb{N}_0}$, where $2^{\mathbb{N}_0}$ denotes the power set of $\mathbb{N}_0$ and $0 < |d(a)| < \infty$ for all $a \in A$. For each arc $a = uv \in A$ the set $d(a)$ is the set of all possible traversal times from u to v (for trains think of all possible running times from a station u to a station v, *e. g.*, we may allow a train to drive fast or slow).

traversal-time-function d

Definition 2.2 Let $G = (V, A)$ be a directed graph with loops, $d\colon A \to 2^{\mathbb{N}_0}$ a traversal-time-function with

- $0 < |d(a)| < \infty$,
- $\forall\, uu \in A\colon d(a) = \{1\}$,

and time steps $T = \{1, \dots, T^{\max}\}$. The *time expansion of* G is the network $G^T = (V^T, A^T)$ with

$$V^T := V \times T,$$
$$A^T := \{(u, t_u)(v, t_v) \in V^T \times V^T\colon uv \in A, t_v - t_u \in d(uv)\}.$$

time expansion $G^T = (V^T, A^T)$

Time expanded networks are the fundamental building part of our models. We want to use the time expanded networks as described above to construct networks that can model timetables for our trains. Unfortunately the basic route graphs $\bar{G}_r = (\bar{V}_r, \bar{A}_r)$ do not carry enough structure to model the timetable with all requested aspects. In particular, the route graphs do not encode at which station a train is allowed to (or has to) stop and wait. But this information is required because we want to apply different running times and constraints depending on the running behaviour of a train. For this we define the following *base train graph* $G_r = (V_r, A_r)$ for $r \in R$, that contains for each station $u \in \bar{V}_r$ up to two nodes $(u, Stop)$ and (u, Run) encoding whether the train stops at u or passes through.

Definition 2.3 The *base train graph* $G_r = (V_r, A_r)$ of a train $r \in R$ is given by

$$\begin{aligned} V_r :=& (\{u \in \bar{V}_r\colon r \text{ may stop at } u\} \times \{Stop\}) \\ & \dot\cup (\{u \in \bar{V}_r\colon r \text{ may pass through } u\} \times \{Run\}), \\ A_r :=& \{(u, b_u)(v, b_v) \in V_r \times V_r\colon uv \in \bar{A}_r\} \dot\cup \{(u, Stop)(u, Stop)\colon (u, Stop) \in V_r\}. \end{aligned}$$

The arcs $(u, Stop)(u, Stop) \in A_r$ are called *wait arcs*.

$G_r = (V_r, A_r)$

The criteria, which train may stop at or pass through a station, depend on the problem at hand. We assume that this information is given by the data and incorporated in the base train graphs. An example of a simple base train graph is shown in Figure 2.2.

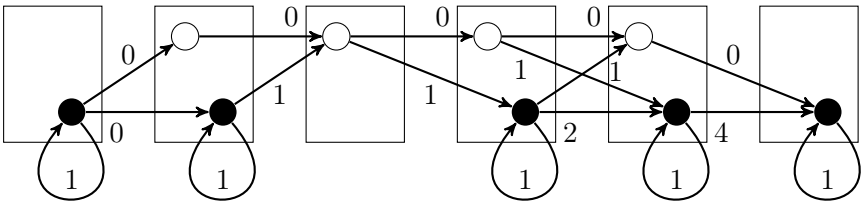


Figure 2.2: Base train graph. The numbers denote the (single) discretised running times. Each "station" corresponds to a box with up to two nodes. The black nodes are *Stop*-nodes that allow waiting and thus have a loop, the white nodes are *Run*-nodes without the possibility to wait.

Next we have to define the traversal time functions $d_r \colon A_r \to 2^{\mathbb{N}_0}$, $r \in R$, for the base train graphs. These traversal times will resemble the running times of the train. Because our model only uses discretised times we have to round the running times appropriately. The simplest approach is to round them to the nearest "integral" time steps as follows. Let $\underline{t}_r^u$ denote the fastest possible running time of $r \in R$ from u_1^r to $u = u_k^r$ for $k \in \{1, \dots, |\bar{V}_r|\}$ and denote by $[t]_T$ the nearest time index for a time t (*e. g.*, if t is in seconds and the time steps T are minutes, then $[t]_T \in T$ denotes the nearest minute). Then the (rounded) traversal times are

$$d_r((u, b_u)(v, b_v)) = \begin{cases} 1, & \text{if } u = v \text{ and } b_u = b_v = \mathit{Stop}, \\ \left[\underline{t}_r^u + t_{uv}^R(\theta(r), (b_u, b_v)) + \delta_r^v\right]_T - [\underline{t}_r^u]_T, & \text{if } u \neq v \text{ and } b_v = \mathit{Stop}, \\ \left[\underline{t}_r^u + t_{uv}^R(\theta(r), (b_u, b_v))\right]_T - [\underline{t}_r^u]_T, & \text{otherwise.} \end{cases} \tag{2.1}$$

d_r

Note that a transfer arc to a node v where the train must stop, *i. e.* has a non-zero minimal stopping time, incorporates the minimal stopping time in addition to the running time itself. Thus such an arc represents both, the transfer from u to v and the subsequent waiting at v.

Now we are ready to describe the time expanded network of a graph.

Definition 2.4 The *train graph* of $r \in R$ is the time expanded network $G_r^T = (V_r^T, A_r^T)$ obtained by time expansion of $G_r = (V_r, A_r)$ with time steps $T = \{1, \dots, T^{\max}\}$ and traversal time function d_r.

$G_r^T = (V_r^T, A_r^T)$

Figure 2.3 shows an example of a time expansion. A timetable of a train corresponds to

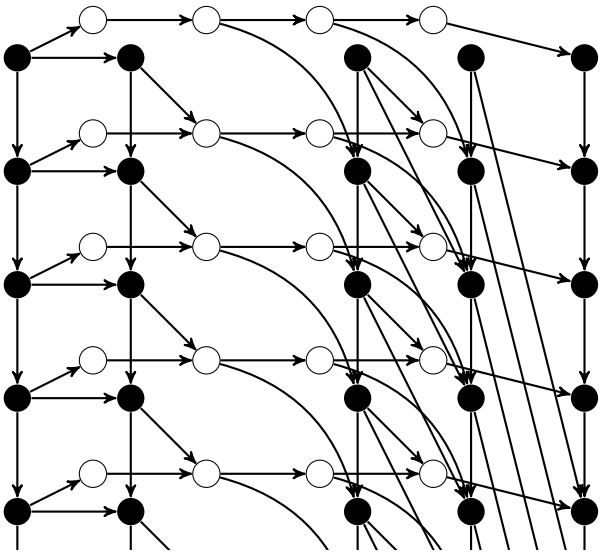


Figure 2.3: The time expansion $G_r^T = (V_r^T, A_r^T)$ of the graph in Figure 2.2.

a path from a node $((u_1, b_1), I_r^0) \in V_r$ to some node $((u_{|\bar{V}_r|}, b_{|\bar{V}_r|}), t) \in V_r$ for an arbitrary arrival time $t \in T$ at the final destination $u^r_{|\bar{V}_r|}$. By construction of the time expanded network G_r^T such a path contains for each $u_i \in \bar{V}_r$ a first node $((u_i, b_i), \underline{t}_i)$ and a last node
$\underline{t}_i$ $((u_i, b_i), \bar{t}_i)$ with $\underline{t}_i \leq \bar{t}_i$. The time $\underline{t}_i$ is then the *arrival time* (which includes the minimal
$\bar{t}_i$ stopping time, see above) of r at station u_i and $\bar{t}_i$ the *departure time* of r at station u_i. The correspondence between paths in G_r^T and timetables naturally leads to a path based modelling approach. We denote by

$$\mathcal{P}_r := \{P \colon P \text{ is a } ((u_1, b_1), I_r^0)\text{-}((u_{|\bar{V}_r|}, b_{|\bar{V}_r|}), t)\text{-path in } G_r^T\}, r \in R,$$

the set of all feasible paths respective timetables of train $r \in R$. With a slight abuse of notation we will write

$$x_r = (x_{r,e})_{e \in A_r^T} \in \mathcal{P}_r$$

for x_r to be the characteristic vector of a path in $\mathcal{P}_r$ if it is clear from the context that x_r is a vector of binary variables.

Because of the lack of a clear operational objective, *e. g.* some financial revenue, we use a rather artificial objective function that is designed with the operational rules in mind that have been described at the end of Section 2.2.

2.4 Formulation as Integer Program

Using the formulations above we can state the problem as an integer program. We introduce one binary variable $x_{r,e} \in \{0,1\}$ for each arc $e \in A_r^T$ in a train graph G_r^T, $r \in R$.

Problem (Basic TTP)

$$\text{minimise} \quad \sum_{r \in R} \sum_{e \in A_r^T} w_{r,e} \cdot x_{r,e} \tag{2.2}$$

$$\text{subject to} \quad x = (x_r)_{r \in R} \in \bigtimes_{r \in R} \mathcal{P}_r, \tag{2.3}$$

departure constraints,

capacity constraints,

headway constraints.

The objective function (2.2) simply adds the cost of each single arc. The concrete values of w_e, $e \in A_r^T$, $r \in R$, are described in Section 2.7 below. The constraints (2.3) mean that the variables x_r, $r \in R$, should correspond to the characteristic vector of a feasible path in the train graph G_r^T. We will always write them in this short form. In an integer program they are typically described using *flow conservation constraints*

$$\sum_{e \in \delta^+(p)} x_{r,e} - \sum_{e \in \delta^-(p)} x_{r,e} = \beta_p, \qquad p = ((u, b_u), t_u) \in V_r^T, r \in R, u \neq u^r_{|\bar{V}_r|}, \tag{2.4}$$

where

$$\beta_p = \begin{cases} 1, & p = ((u, b_u), t_u), u = u_1^r, t_u = I_r^0, \\ 0, & \text{otherwise.} \end{cases}$$

The second set of constraints are the *departure constraints* at stopping stations. A train must not leave a stopping station before the beginning of the corresponding stopping interval. We ensure this by simply forbidding the use of leaving transfer arcs with an departure time smaller than the beginning of the stopping interval

$$x_{r,e} = 0, \qquad e = (((u, b_u), t_u), ((v, b_v), t_v))^r \in A_r^T, u \neq v, r \in R, t_u < \underline{I}_r^u. \tag{2.5}$$

Remark 2.5 In practice, the departure constraints (2.5) will not be contained as explicit equality constraints in the model. Instead, one deals with them implicitly in the set of paths $\mathcal{P}_r$, $r \in R$, *i. e.* paths that contain at least one of the forbidden arcs are removed

from this set. This does not cause any difficulties on the subproblems in a certain graph G_r^T. Another, equivalent approach would be to set the objective value of the forbidden arcs to $+\infty$ (or a very high value in practice), so that no optimal path would contain such an arc. Nevertheless, we will keep these constraints in the algebraic description of our models in the following.

The (station) capacity constraints and the headway constraints are *coupling* constraints. The latter need more detailed investigation and we postpone their discussion to the next sections. The former can easily be modelled by linear inequalities as follows. Recall that we have two types of capacities. Absolute capacities denote the maximal number of trains that can visit a certain node at the same time. Directional capacities denote the maximal number of trains that can visit a certain node at the same time *coming from the same direction*. The structure of the inequality constraints is the same in both cases: Only a bounded number of arcs that represent a train arriving at or waiting at a station u may be selected in the train paths simultaneously. For this we collect those arcs in the following sets for all $v \in V^I$, $a = uv \in A^I$ and time steps $t \in T$:

$$\begin{aligned} A^-(v,t) :=& \{e^r = (((u,b_u),t_u),((v,b_v),t))^r : e^r \in A_r^T, r \in R\}, \\ A^-(uv,t) :=& \{e^r = (((u,b_u),t_u),((v,b_v),t))^r : e^r \in A_r^T, r \in R\} \\ & \cup \{e^r = (((v,Stop),t-1),((v,Stop),t))^r : e^r \in A_r^T, uv \in \bar{A}_r, r \in R\}. \end{aligned}$$

Note that in case of directional capacities we have to add the wait-arcs of exactly those trains to $A^-(uv,t)$, that arrive from u. Using these sets the capacity constraints can be stated easily:

$$\sum_{e^r \in A^-(p,t)} x_{r,e} \leq c_p, \qquad p \in V^I \cup A^I, t \in T. \tag{2.6}$$

The last part of the model are the headway constraints. In general there are two approaches in the literature (see Borndörfer and Schlechte [9]) to handle these restrictions in the model.

1. Additional inequalities forbid the simultaneous usage of (transfer) arcs that have a pairwise headway conflict, *i. e.* that represent train runs that are too close to each other.

2. An additional *configuration*-network is used to "open" the track in a conflict free manner for certain train runs.

In our project we tested both approaches, they are described in the following two sections.

2.5 Clique Based Headway Constraints

Headway constraints are always associated with a certain infrastructure arc $a \in A^I$. For the sake of simplicity we assume that the arc is a double-track $a \in A^{I,2}$, in this case trains running in the opposite direction a^{-1} have no interactions with those running on a. In the case of a single track one has, in addition to the headway times between trains running in the same direction, to observe headway times between trains running in opposite directions. But their treatment in the constraints described below is completely analogous (one could think of trains running in the opposite directions as having different train types and appropriate headway times).

In order to represent the exclusive use of arcs in A^I coupled by headway restrictions, we build a conflict graph $G_a^C = (V_a^C, A_a^C)$ in the following way. The nodes correspond to all possible runs of some train over this infrastructure arc $a \in A^I$ $G_a^C = (V_a^C, A_a^C)$

$$V_a^C = \{(((u, b_u), t_u), ((v, b_v), t_v))^r : r \in R, (((u, b_u), t_u), ((v, b_v), t_v))^r \in A_r^T, uv = a\}. \qquad V_a^C$$

Two train runs are in conflict if their respective starting times are too close to each other (compared with the headway time). Two nodes are connected by an arc in the conflict graph if and only if they have a headway conflict

$$\begin{aligned} A_a^C := \Big\{ p^r q^{r'} :\ & p^r = (((u, b_u), t_u), ((v, b_v), t_v))^r \in V_a^C, \\ & q^{r'} = (((u', b'_u), t'_u), ((v', b'_v), t'_v))^{r'} \in V_a^C, \\ & \big[(t_u \le t'_u) \wedge t'_u - t_u < t_a^{HW}(\theta(r), (b_u, b_v), \theta(r'), (b'_u, b'_v))\big] \\ & \vee \big[(t_u \ge t'_u) \wedge t_u - t'_u < t_a^{HW}(\theta(r'), (b'_u, b'_v), \theta(r), (b_u, b_v))\big] \Big\}. \end{aligned} \qquad A_a^C$$

A clique in the conflict graph is a subset of nodes $\mathcal{C} \subseteq V_a^C$ so that each pair of nodes in $\mathcal{C}$ is connected by an arc in G_a^C, *i. e.* $\forall\, p^r, q^{r'} \in \mathcal{C}, p^r \neq q^{r'} : pq \in A_a^C$. Each clique $\mathcal{C}$ in the conflict graph induces an inequality constraint

$$\sum_{e^r \in \mathcal{C}} x_{r,e} \le 1, \qquad \mathcal{C} \text{ clique in } G_a^C, a \in A^I, \tag{2.7}$$

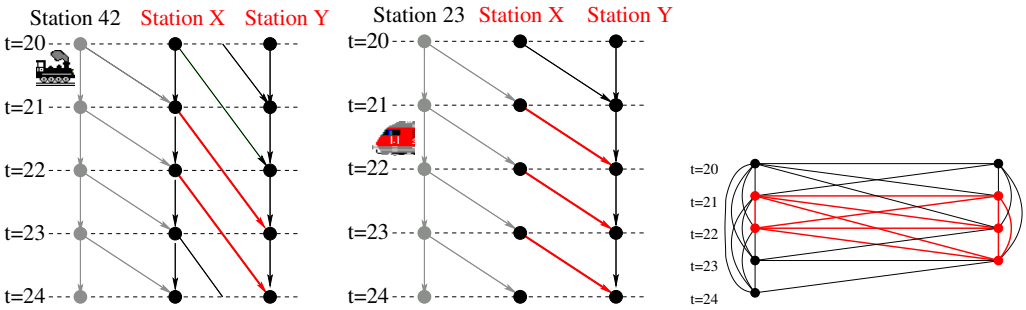
i. e., at most one of the train runs in $\mathcal{C}$ can be used at the same time. Obviously the strongest inequalities are exactly those induced by the *maximal* cliques in G_a^C. But for an IP formulation it would be sufficient to add all inequalities induced by the arcs in G_a^C

$$x_{r_1,e_1} + x_{r_2,e_2} \le 1, \qquad e_1^{r_1} e_2^{r_2} \in A_a^C, a \in A^I,$$

although this leads to a weak LP relaxation. A simple example of clique constraints is shown in Figure 2.4.

The main drawback of clique inequalities is that for a good LP relaxation one would have to add the inequalities induced by the maximal cliques in the conflict graph. Unfortunately finding a maximal clique in a graph is itself a hard problem. In fact, separating the maximal violated inequalities turned out to be very difficult, so in practice one usually restricts to separating small (usually not maximal) cliques, see Section 3.5.1.

The final IP formulation using clique inequalities reads as follows.



(a) Slow train (b) Fast train (c) Conflict graph

Figure 2.4: Illustration of a clique constraint. Two trains run over one infrastructure arc $a \in A^I$, both trains must stop at stations X and Y. If the first (slow) train runs first the headway time is 3 minutes, otherwise it is 2 minutes. The highlighted arcs in both train graphs are in pairwise conflict (note that two runs of the same train are always in conflict because each train must run exactly once) and form a maximal clique in the conflict graph shown in Figure 2.4c.

Problem (Clique Formulation)

(TTP-clq)

$$\begin{aligned}
&\text{minimise} && \sum_{r \in R} \sum_{e \in A_r^T} w_{r,e} \cdot x_r && \\
&\text{subject to} && x = (x_r)_{r \in R} && \in \bigtimes_{r \in R} \mathcal{P}_r, \\
& && x_{r,e} = 0, && e^r = (((u, b_u), t_u), ((v, b_v), t_v))^r \in A_r^T, \\
& && && u \neq v, r \in R, t_u < \underline{I}_r^u, \\
& && \sum_{e^r \in A^-(p,t)} x_{r,e} \leq c_p, && p \in V^I \cup A^I, t \in T, \\
& && \sum_{e^r \in \mathcal{C}} x_{r,e} \leq 1, && \mathcal{C} \text{ clique in } G_a^C, a \in A^I,
\end{aligned}$$

2.6 Configuration Networks

Another approach for modelling headway constraints has been introduced by Borndörfer and Schlechte [9]. The idea is as follows. Each feasible set of train runs over a given infrastructure arc, *i. e.* runs that do not have headway conflicts for this arc, is called a *configuration*. The set of all possible configurations can then be modelled by a *configuration network*, which is a time expanded network similar to the train graphs, and a (feasible) configuration corresponds to a certain path in this network.

As before we consider a single fixed infrastructure arc $a \in A^{I,2}$ that corresponds to a double track. Single tracks can be handled accordingly. Formally, a configuration network contains exactly one *configuration arc* for each train run over the arc a, these are connected by *headway arcs* that ensure the observance of the headway restrictions. Let σ_a and τ_a denote an artificial start resp. terminal node, then the configuration network $G_a^T = (V_a^T, A_a^T)$ is defined by $G_a^T = (V_a^T, A_a^T)$

$$
\begin{aligned}
V_a^T = {} & \{\sigma_a, \tau_a\} \cup \{e^+, e^- \colon e = (((u, b_u), t_u), ((v, b_v), t_v))^r \in A_r^T, uv = a\}, \\
A_a^T = {} & \{\sigma_a \tau_a\} \cup \{\sigma_a e^+ \colon e^+ \in V_a^T\} \cup \{e^- \tau_a \colon e^- \in V_a^T\} \\
& \cup \{e^+ e^- \colon e^+, e^- \in V_a^T\} \\
& \cup \{e^+ f^+ \colon e = (((u, b_u), t_u), ((v, b_v), t_v))^r, \\
& \qquad f = (((u, b_u), t_u + 1), ((v, b_v), t_v + 1))^r\} \\
& \cup \{e^- f^+ \colon e = (((u, b_u), t_u), ((v, b_v), t_v))^r, \\
& \qquad f = (((u', b'_u), t'_u), ((v', b'_v), t'_v))^{r'}, \\
& \qquad r \neq r', \\
& \qquad t'_u - t_u = \lceil t^{HW}(\theta(r), (b_u, b_v), \theta(r'), (b'_u, b'_v)) \rceil_T\}.
\end{aligned}
$$

Although these definitions look weird they are actually quite natural.

- The two artificial nodes σ_a and τ_a are connected with respective start and end nodes e^+ or e^-.

- Each transfer arc $e = (((u, b_u), t_u), ((v, b_v), t_v))^r$ corresponds to exactly two nodes e^+, e^- and one *configuration arc* e^+e^- in the configuration network (one can think of e^+e^- as a copy of e).

- If a train run $e = (((u, b_u), t_u), ((v, b_v), t_v))^r$ is followed immediately by a run $f = (((u', b'_u), t'_u), ((v', b'_v), t'_v))^{r'}$ so that the difference of their respective start times t_u and t'_u equals the headway time, then these two runs can be contained (in that order) in a valid configuration. We connect the end node e^- with the start node f^+ of those two runs.

- The distance between two successive runs may actually be larger than the headway time. This means that the configuration can "wait" before the next transfer happens. This is modelled by configuration-wait-arcs e^+f^+ where e and f are equivalent runs so that f happens one time step after e, *i. e.* $e = (((u, b_u), t_u), ((v, b_v), t_v))^r$ and $f = (((u, b_u), t_u + 1), ((v, b_v), t_v + 1))^r$.

A feasible configuration corresponds to a σ_a-τ_a-path in G_a^T that contains *exactly one* run for each train.

Figure 2.5 shows an example configuration network for two trains.

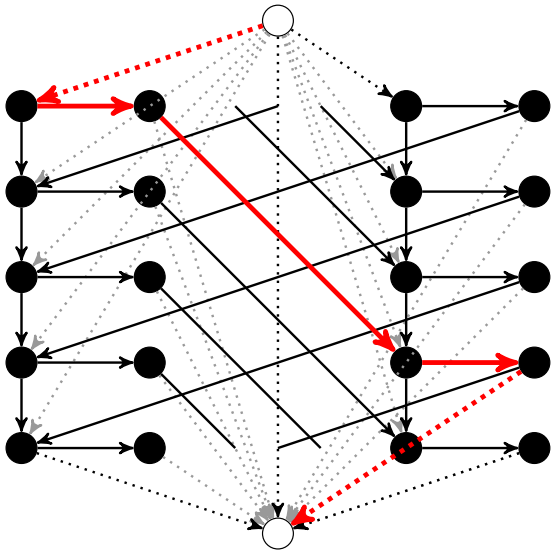


Figure 2.5: Configuration network.

The configuration networks are modelled analogously to the train graphs using flow conservation constraints. Again we use the short notation

$$\mathcal{P}_a := \{P\colon P \text{ is a } \sigma_a\text{-}\tau_a\text{-path in } G_a^T \text{ and} \\ \text{contains exactly one configuration arc for each contained train}\}, \quad a \in A^I,$$

for the set of all paths that correspond to a feasible configuration, and write

$$x_a \in \mathcal{P}_a, \qquad a \in A^I,$$

for x^a being the characteristic vector of such a path. Finally we add simple coupling constraints that allow the usage of a certain train transfer arc if and only if the corresponding configuration arc is used in the configuration network. For a transfer arc $e = (((u, b_u), t_u), ((v, b_v), t_v))^r \in A_r^T$, $r \in R$, $a = uv \in A^I$, we denote by $\mathrm{cfg}(e) = e^+e^- \in G_a^T$ its corresponding configuration arc. Then the *configuration constraints* are the equality constraints

$\mathrm{cfg}(e)$

$$x_{r,e} = x_{a,\mathrm{cfg}(e)}, \qquad r \in R, a = uv \in A^I, e^r = (((u, b_u), t_u), ((v, b_v), t_v))^r \in A_r^T. \tag{2.8}$$

Although configuration networks provide stronger relaxations than clique inequalities [9], they still have the drawback that their complete description is too difficult to solve, because the requirement to contain exactly one configuration arc for each train leads to a constraint shortest path problem. In practice one usually relaxes this condition and uses the following set of paths

$$\tilde{\mathcal{P}}_a := \{P\colon P \text{ is a } \sigma_a\text{-}\tau_a\text{-path in } G_a^T\} \supseteq \mathcal{P}_a, \qquad a \in A^I.$$

This simpler set contains more paths and each path is a feasible configuration in the sense that corresponding train runs do not have headway conflicts. However, it also contains paths that contain more than one configuration arc for a certain train $r \in R$. Such a path can never be contained in a feasible solution of the problem, because the train can only use the transition arc associated with one of those configuration arcs. The configuration constraints $x_{r,e} = x_{a,\mathrm{cfg}(e)}$ for the other configuration arcs would therefore be violated. This implies that using $\tilde{\mathcal{P}}_a$ instead of $\mathcal{P}_a$ also gives a valid IP formulation of the TTP, but the relaxation may be weaker. In all our implementations we always used the weaker formulation with $\tilde{\mathcal{P}}_a$, although we will write $\mathcal{P}_a$ for simplicity.

The final IP formulation using configuration networks reads

Problem (Configuration Formulation)

(TTP-cfg)

$$
\begin{aligned}
\text{minimise} \quad & \sum_{r \in R} \sum_{e \in A_r^T} x_{r,e} \cdot x_{r,e} && \\
\text{subject to} \quad & x = (x_p)_{p \in R \cup A^I} \in \bigtimes_{p \in R \cup A^I} \mathcal{P}_p, && \\
& x_{r,e} = 0, && e = ((u, b_u), t_u)((v, b_v), t_v)^r \in A_r^T, \\
& && u \neq v, r \in R, t_u < \underline{I}_r^u, \\
& \sum_{e \in A^-(p)} x_e \leq c_p, && p \in V^I \cup A^I, t \in T, \\
& x_{r,e} = x_{uv,\mathrm{cfg}(e)}, && r \in R, uv \in A^I, \\
& && e = ((u, b_u), t_u)((v, b_v), t_v)^r \in A_r^T.
\end{aligned}
$$

In most of our tests we will use the clique based formulation because it behaved algorithmically better than the configuration based formulation for our solution approach. We compare both approaches in Section 6.4.3.

2.7 Objective Function

In this section we discuss the objective function that we use for our TTP model. The objective function is defined via arc weights in the IP-model, *i. e.*, each arc that is contained in some solution path associated with a train contributes a certain amount to the overall objective value. As mentioned earlier, this objective function is completely artificial, *i. e.*, it is not based on any cost or profit or energy measurements. Instead it is designed with the following goals in mind:

- Passenger trains $r \in R$, $\theta(r) \in \Theta^p$, should reach *each* stopping station without a delay. In particular, if r reaches a station $u \in \bar{V}_r$ with $I_r^u = [\underline{I}_r^u, \bar{I}_r^u] \neq [-\infty, +\infty]$ at a time $t \in T$ with $t > \bar{I}_r^u$, then a penalty value of $F_{r,u}(t - \bar{I}_r^u)$ (defined below) must

be paid. Note that a delay at some station u may propagate to a later station v. Then the penalty is paid at each station unless the train can run fast enough to catch up.

- The delay of a long distance train is more expensive than that of a short distance train.
- Freight trains should reach their final destination as early as possible. As a minor objective, they should also prefer routes that reach intermediate stations early.
- Delay should be penalised progressively: the first minute of delay is relatively cheap while the second minute is more expensive, the third minute even more and so on.

We start with the objective function of passenger trains.

Passenger trains. In this paragraph we will always assume that $r \in R$ is a passenger train, *i. e.* $\theta(r) \in \Theta^p$. One important aspect of the objective is that trains have a different number of stopping stations. In particular, a short distance train stopping at many small regional stations may have more stopping stations than a long distance train that stops only at few major stations. Therefore the concrete penalty value should not only depend on the delay and the train type but should also take the number of stations a train visits as well as the length of the train run into account.

This motivates the following definition of the objective function. Let $P = u_i \dots u_j \subset \bar{G}_r$ be a path in the base graph, then we denote by

$\underline{d}_r(P)$
$$\underline{d}_r(P) := \min \left\{ \sum_{k=i}^{j-1} \min d_r((u_k, b_k)(u_{k+1}, b_{k+1})) \colon (u_l, b_l) \in \bar{V}_r, l \in \{i, \dots, j\} \right\}$$

the running time of the fastest possible run along P. Furthermore let

$\bar{V}_r^S$
$$\bar{V}_r^S := \{u \in \bar{V}_r \colon I_r^u \neq [-\infty, \infty]\}$$

denote the set of all stopping stations of train r. For each $v \in \bar{V}_r^S$ we denote by $P_r^v \subset \bar{G}_r$ the route from the preceding stopping station $u \in \bar{V}_r^S$. In particular, if $v = u_j \in \bar{V}_r$ and there is a node $u = u_i \in \bar{V}_r$ with $i < j$ so that $u_k \notin \bar{V}_r^S$ for all k, $i < k < j$, then $P_r^v = u_i \dots u_j$, otherwise v is the first stopping station and we set $P_r^v = u_1 \dots v$. The length of the section before a stopping station $u \in \bar{V}_r^S$ is then the fastest run from the preceding stopping station

ℓ_r^u
$$\ell_r^u := \underline{d}_r(P_r^u),$$

and the length of the whole train run is

ℓ_r
$$\ell_r := \underline{d}_r(P), P = u_1 \dots u_{|\bar{V}_r|} \subset \bar{G}_r.$$

We choose a penalty value of a delay at a certain station that is proportional to the length of the preceding section. As mentioned above, the penalty value should be progressive,
α_θ *i. e.*, it should increase super linearly with increasing delay. Let $\alpha_\theta > 0$ be the weight

factor associated with train type $\theta \in \Theta$ and $\delta \in \mathbb{N}_0$ a certain delay in time steps, then we set the penalty function for $r \in R$ and $u \in \bar{V}_r^S$

$$F_{r,u}\colon \mathbb{N}_0 \to \mathbb{R}, \qquad F_{r,u}$$
$$F_{r,u}(\delta) := \alpha_{\theta(r)} \cdot \frac{\ell_r^u}{\ell_r} \cdot \delta^2.$$

The objective value depends on the delay of the train when it arrives at a certain stopping station. Thus we assign non-zero costs only to the transfer arcs whose sink station is a stopping station. For an arc $e^r = (((u, b_u), t_u), ((v, b_v), t_v))^r \in A_r^T$, $r \in R$, we set

$$w_{r,e} := \begin{cases} 0, & \text{if } u = v, \\ 0, & \text{if } u \neq v \notin \bar{V}_r^S, \\ F_{r,v}(\max\{0, t_v - \delta_r^v - \bar{I}_r\}), & \text{otherwise.} \end{cases} \qquad (2.9) \qquad w_{r,e}$$

(Note that t_v includes the minimal stopping time at v, see (2.1), so we subtract δ_r^v to get the arrival time at v.)

Freight trains. Freight trains differ from passenger trains in the objective that only the arrival time at the final station is significant. Furthermore freight trains do not have any intermediate forced stopping stations (although that could be covered by the model, *e. g.*, if a few additional waggons are appended to a train in a certain station). Consequently the objective function of a freight train has the same structure as the objective function of a passenger train with the set of "stopping stations" set to $\bar{V}_r^S := \{u_{|\bar{V}_r|}\}$.

In fact, we will use a slight variation of this objective function. The main part will be as described above, penalising only the arrival at the final station. In addition we will add a further, smaller objective term to each intermediate station so that freight trains tend to run as early as possible, too. This corresponds to the objective function for passenger trains, this time choosing $\bar{V}_r^S = \bar{V}_r$, *i. e.* each station is considered as stopping station. The intuition is the overall *minor* objective, that trains should start as early as possible, without interfering with other trains too much, so that in the case of a delay (in practice, when the generated timetable is used, not in the optimisation model) sufficient additional buffer time is available for compensation.

Putting all together we define

$$\ell_r^v := \begin{cases} 0, & \text{if } v = u_1 \in \bar{V}_r, \\ \underline{d}_r(uv), & \text{otherwise with } u \in \bar{V}_r\colon uv \in \bar{A}_r, \end{cases}$$

and

$$\underline{t}_r^u := I_r^0 + \underline{d}_r(u_1 \dots u),$$

where $\underline{t}_r^u$ is the earliest possible arrival time of r at station $u \in \bar{V}_r$. We use the following two penalty functions

$F_r^1, F_{r,u}^2$ $$F_r^1, F_{r,u}^2 \colon \mathbb{N}_0 \to \mathbb{R},$$

$F_r^1(\delta)$ $$F_r^1(\delta) := \alpha^1_{\theta(r)} \cdot \delta^2,$$

$F_{r,u}^2(\delta)$ $$F_{r,u}^2(\delta) := \alpha^2_{\theta(r)} \cdot \frac{\ell_r^u}{\ell_r} \cdot \delta^2.$$

As for passenger trains, we assign non-zero costs only to transfer arcs, independent of whether the sink node is a stopping station. Let $e^r = (((u, b_u), t_u), ((v, b_v), t_v))^r \in A_r^T$, $r \in R$, we set

$$w_{r,e} := \begin{cases} 0, & \text{if } u = v, \\ F_{r,v}^2(t_v - \underline{t}_r^v), & \text{if } u \neq v \neq u_{|\bar{V}_r|} \in \bar{V}_r, \\ F_r^1(t_v - \underline{t}_r^v) + F_{r,v}^2(t_v - \underline{t}_r^v), & \text{if } u \neq v = u_{|\bar{V}_r|} \in \bar{V}_r. \end{cases} \tag{2.10}$$

Penalty weights. The concrete values for the penalty weights α_θ, α_θ^1 and α_θ^2 are chosen with the following hierarchy in mind:

1. long distance passenger trains,
2. short distance passenger trains,
3. freight trains.

We assume that this hierarchy holds on the whole network. In certain applications there may also be some track on which freight trains have a preference over long distance trains, but we do not take such considerations into account.

The concrete values for the weights are artificial but chosen so that roughly

- one minute (time step) delay of a long distance train is worth about 30 minutes delay of a freight train,
- one minute (time step) delay of a short distance train is worth about 20 minutes delay of a freight train.

Thus we set for $\theta \in \Theta^p$

$$\alpha_\theta = \begin{cases} 30, & \text{if } \theta \in \Theta^{\mathrm{ld}}, \\ 20, & \text{if } \theta \in \Theta^{\mathrm{sd}}, \end{cases}$$

and for freight trains $\theta \in \Theta^f$

$$\alpha_\theta^1 = 1, \qquad \alpha_\theta^2 = 0.01.$$

Remark 2.6 The objective function described in this section may lead to many equivalent solutions. For example, the objective function of passenger trains only depends on the arrival time at stopping stations, whereas the arrival at other intermediate stations has no influence. In order to make the solution process more robust w. r. t. these symmetries, we usually add very small artificial costs to other arcs. (For example, the Lagrangian relaxation based solution process, which will be described in Chapter 3, should always return the same, hopefully unique, shortest path from a subproblem. Small perturbation costs increase the probability of this uniqueness, so the returned path should, *e. g.*, be independent from the concretely employed shortest path algorithm. The uniqueness of solutions is not strictly necessary from a theoretical point of view, but is quite convenient in practice.)

3 Solution Methods

3.1 Introduction

In this chapter we will describe the basic solution methods that we employed for our train timetabling problem. We will not discuss the whole solution process but focus on one important aspect, the solution of the relaxation. As stated before, the purpose of this work is to improve the understanding of this particular step and to develop new tools and methods to enhance the solution process. We start with a generic problem formulation in Section 3.2 (the TTP is a special case). In Section 3.3 we describe our main solution approach, Lagrangian relaxation, and in Section 3.4 another often used solution approach, Column Generation. Afterwards we discuss first order methods as a general algorithmic approach for solving Lagrangian relaxation in Section 3.5 and we close this chapter with a description of Lagrangian relaxation in the context of the TTP.

3.2 Problem Formulation

We consider a general Combinatorial Optimisation Problem in the following form

Problem Let R be a finite set and for each $r \in R$

- $\mathcal{X} = \bigtimes_{r\in R} \mathcal{X}_r$ with $\mathcal{X}_r \subseteq \mathbb{R}^{n_r}$, $n_r \in \mathbb{N}$, compact *ground sets*,
- $h_r \colon \mathcal{X}_r \to \mathbb{R}$ linear *primal objective functions*,

and set $n = \sum_{r\in R} n_r$. Furthermore let $M = \mathcal{E} \dot\cup \mathcal{I}$ be a (finite) set and

- $C \in \mathbb{R}^{M,n}$,
- $b \in \mathbb{R}^M$.

Then we consider a *combinatorial optimisation problem* of the following form.

$$
\text{(CP)} \qquad \begin{aligned}
\text{maximise} \quad & \sum_{r\in R} h_r(x_r) \\
\text{subject to} \quad & x_r \in \mathcal{X}_r, && r \in R, \\
& \sum_{r\in R} C_{\mathcal{E},r} x_r = b_{\mathcal{E}}, \\
& \sum_{r\in R} C_{\mathcal{I},r} x_r \le b_{\mathcal{I}}.
\end{aligned}
$$

$v(CP)$

The equalities $\sum_{r\in R} C_{\mathcal{E},r}x_r = b_{\mathcal{E}}$ and inequalities $\sum_{r\in R} C_{\mathcal{I},r}x_r \leq b_{\mathcal{I}}$ are called *coupling constraints*. The *optimal value* of (CP) is denoted by $v(CP)$.

If each ground set $\mathcal{X}_r$, $r \in R$, consists of the integer points within a polyhedron the problem is called an *Integer Program* (IP). Whenever the sets $\mathcal{X}_r$ are non-convex then these problems are in general very difficult. The standard algorithmic approach is to solve a *relaxation* of this problem.

Depending on the structure of the ground sets $\mathcal{X}_r$, $r \in R$, they may be written in the form $\mathcal{X}_r = \mathbb{Z}^{n_r} \cap \{x_r \in \mathbb{R}^{n_r} : \bar{C}_r x_r \leq \bar{b}^r\}$ for some $\bar{C}_r^r \in \mathbb{R}^{m_r \times n_r}$, *i. e.* as the intersection of a polyhedron and the integer lattice $\mathbb{Z}^{n_r}$. In this case an obvious relaxation is to neglect the integrality condition $\mathcal{X}_r \subseteq \mathbb{Z}^{n_r}$ and replace $\mathcal{X}_r$ with $\{x_r \in \mathbb{R}^{n_r} : \bar{C}_r x_r \leq \bar{b}^r\}$, which leads to the following *Linear Programming Relaxation*

Problem (Linear Programming Relaxation) Assume

$$\mathcal{X}_r = \mathbb{Z}^{n_r} \cap \{x_r \in \mathbb{R}^{n_r} : \bar{C}_r x_r \leq \bar{b}^r\}$$

for all $r \in R$. The *Linear Programming Relaxation* of (CP) is

(LP)
$$\begin{aligned} \text{maximise} \quad & \sum_{r\in R} h_r(x_r) \\ \text{subject to} \quad & \sum_{r\in R} \bar{C}_r x_r \leq \bar{b}^r, \qquad r \in R, \\ & \sum_{r\in R} C_{\mathcal{E},r} x_r = b_{\mathcal{E}}, \\ & \sum_{r\in R} C_{\mathcal{I},r} x_r \leq b_{\mathcal{I}}. \end{aligned}$$

This relaxation is a standard linear optimisation problem and can be solved efficiently by state-of-the-art interior point algorithms (see, *e. g.*, Roos et al. [83]) and for many practical applications by the simplex method (see, *e. g.*, Vanderbei [93]). For very large scale instances, however, these linear relaxations may be quite large and their practical solution by standard solvers becomes challenging even today.

3.3 Lagrangian Relaxation

Another successful relaxation approach is *Lagrangian relaxation*, see, *e. g.*, Lemaréchal [64] and the references therein for an overview. It is typically applied if the hardness of the original optimisation problem (CP) is caused by the coupling constraints. In other words, we assume that the *subproblems* for each $r \in R$

Problem (Subproblem)

$$(Sub^r(c_r)) \qquad \begin{aligned} \text{maximise} \quad & h_r(x_r) + \langle c_r, x_r \rangle \\ \text{subject to} \quad & x_r \in \mathcal{X}_r, \end{aligned}$$

can be solved efficiently for each linear augmenting term $c_r \in \mathbb{R}^{n_r}$ by some specific algorithm. We denote the optimal value by $v(Sub^r(c_r))$ (note that this value depends on c_r).

The idea of Lagrangian relaxation is, instead of enforcing the coupling constraints, to penalise their violation in the objective function. The *Lagrangian* $L(x, y)$ is defined by

$$\begin{aligned} L\colon \mathcal{X} \times \mathbb{R}^M &\to \mathbb{R}, \\ L(x, y) &= \sum_{r \in R} h_r(x_r) + \langle y, b - Cx \rangle. \end{aligned}$$

The variables y are called *Lagrange multipliers*. The original problem (CP) is equivalent to

$$\begin{aligned} \text{maximise} \quad & \inf\{L(x, y)\colon y = (y_{\mathcal{E}}, y_{\mathcal{I}}) \in \mathbb{R}^{\mathcal{E}} \times \mathbb{R}^{\mathcal{I}}_+\} \\ \text{subject to} \quad & x_r \in \mathcal{X}_r, \quad r \in R. \end{aligned}$$

Furthermore, if we exchange the max and the inf in this formulation we get an upper bound on the objective value (so called *weak duality*, Hiriart-Urruty and Lemaréchal [55]). This problem is called the *Lagrangian relaxation* of (CP)

Problem (Lagrangian relaxation)

$$(\text{LD}) \qquad \begin{aligned} \text{minimise} \quad & \max\{L(x, y)\colon x \in \mathcal{X}\} \\ \text{subject to} \quad & y \in \mathbb{R}^{\mathcal{E}} \times \mathbb{R}^{\mathcal{I}}_+, \end{aligned}$$

or, equivalent,

$$\begin{aligned} \text{minimise} \quad & f(y) \\ \text{subject to} \quad & y \in \mathbb{R}^{\mathcal{E}} \times \mathbb{R}^{\mathcal{I}}_+, \end{aligned}$$

where

$$f(y) := \max\{L(x, y)\colon x \in \mathcal{X}\},$$

and it holds $v(CP) \leq v(LD)$. In general, equality (so called *strong duality*) between those two optimal values does not hold. In fact, one can show that $v(LD)$ equals the optimal value of the *convexified* version (conv CP) of the original problem below.

Definition 3.1 Let $h\colon X \to \mathbb{R}$ be an arbitrary function. The *concave hull* of h is the function $(\operatorname{conc} h)$ defined by

$$(\operatorname{conc} h)\colon \operatorname{conv} X \to \mathbb{R},$$
$$\operatorname{epi}(-(\operatorname{conc} h)) = \overline{\operatorname{conv}} \operatorname{epi}(-h),$$

where $\overline{\operatorname{conv}} S$ denotes the closed convex hull of a set S and $\operatorname{epi} h$ the epigraph of h

$$\operatorname{epi} h := \{(x, z)\colon z \geq h(x), x \in X\}.$$

It can be shown that the concave hull is well-defined (Hiriart-Urruty and Lemaréchal [55, Chapter X]). The convexified problem is then

Problem (Convex Hull of (CP))

$$\text{(conv CP)} \qquad \begin{aligned} \text{maximise} \quad & \sum_{r \in R} (\operatorname{conc} h_r)(x_r) \\ \text{subject to} \quad & x_r \in \operatorname{conv} \mathcal{X}_r, \qquad r \in R, \\ & \sum_{r \in R} C_{\mathcal{E},r} x_r = b_{\mathcal{E}}, \\ & \sum_{r \in R} C_{\mathcal{I},r} x_r \leq b_{\mathcal{I}}. \end{aligned}$$

As said before $v(\operatorname{conv} CP) = v(LD)$ holds under mild assumptions, which are met in our case, see Lemaréchal and Renaud [63, theorems 2.11 and 2.12].

3.4 Column Generation

Another approach that has been applied successfully to many optimisation problems is column generation. For an introduction to column generation see Desaulniers et al. [27], in particular, Chapter 1 [28], and Lübbecke and Desrosiers [70]. An example for column generation in time discretised network models for the TTP can be found in Cacchiani et al. [13]. For column generation we assume that the sets $\mathcal{X}_r$, $r \in R$, are discrete and finite, *i. e.*

$$\mathcal{X}_r = \{x_r^1, \ldots, x_r^{\bar{n}_r}\}, \bar{n}_r \in \mathbb{N},$$

and that all primal objective functions are linear

$$h_r(x_r) = \langle w_r, x_r \rangle, r \in R, w_r \in \mathbb{R}^{n_r}$$

(column generation can be applied to more general settings, but this is sufficient for our purposes). In column generation one employs the *Dantzig-Wolfe*-decomposition [24] by using a different set of variables compared to the original problem (CP): for each point $x_r^i \in \mathcal{X}_r$, $r \in R$, $i \in \{1, \dots, \bar{n}_r\}$, we introduce one binary variable $\lambda_r^i \in \{0, 1\}$. Then

$$\mathcal{X}_r = \left\{ \sum_{i=1}^{\bar{n}_r} \lambda_r^i x_r^i \colon \lambda_r^i \in \{0,1\}, \sum_{i=1}^{\bar{n}_r} \lambda_r^i = 1 \right\}$$

and

$$\operatorname{conv} \mathcal{X}_r = \left\{ \sum_{i=1}^{\bar{n}_r} \lambda_r^i x_r^i \colon \lambda_r^i \in [0,1], \sum_{i=1}^{\bar{n}_r} \lambda_r^i = 1 \right\}.$$

These simple relations lead to the following equivalent description of (CP)

$$\begin{aligned}
\text{maximise} \quad & \sum_{r \in R} \sum_{i=1}^{\bar{n}_r} \langle w_r, x_r^i \rangle \lambda_r^i \\
\text{subject to} \quad & \sum_{r \in R} \sum_{i=1}^{\bar{n}_r} C_{\mathcal{E},r} x_r^i \lambda_r^i = b_{\mathcal{E}}, \\
& \sum_{r \in R} \sum_{i=1}^{\bar{n}_r} C_{\mathcal{I},r} x_r^i \lambda_r^i \leq b_{\mathcal{I}}, \\
& \sum_{i=1}^{\bar{n}_r} \lambda_r^i = 1, && r \in R, \\
& \lambda_r^i \in \{0,1\}, && r \in R, i \in \{1, \dots, \bar{n}_r\},
\end{aligned}$$

and its LP-relaxation by using $\operatorname{conv} \mathcal{X}_r$ instead of $\mathcal{X}_r$

$$\begin{aligned}
\text{maximise} \quad & \sum_{r \in R} \sum_{i=1}^{\bar{n}_r} \langle w_r, x_r^i \rangle \lambda_r^i \\
\text{subject to} \quad & \sum_{r \in R} \sum_{i=1}^{\bar{n}_r} C_{\mathcal{E},r} x_r^i \lambda_r^i = b_{\mathcal{E}}, \\
& \sum_{r \in R} \sum_{i=1}^{\bar{n}_r} C_{\mathcal{I},r} x_r^i \lambda_r^i \leq b_{\mathcal{I}}, \\
& \sum_{i=1}^{\bar{n}_r} \lambda_r^i = 1, && r \in R, \\
& \lambda_r^i \geq 0, && r \in R, i \in \{1, \dots, \bar{n}_r\},
\end{aligned}$$

The difficulty with this formulation is that the sets $\mathcal{X}_r$, $r \in R$, are usually very large though finite, so the formulation is very large as well. The idea is to allow only a small number of coefficients λ_r^i to be non-zero, *i. e.*, we choose sets $\tilde{I}_r \subseteq \{1, \dots, \bar{n}_r\}$ and fix

$\lambda_r^i = 0$ for all $i \notin \tilde{I}_r$:

$$
\text{(RMP)} \qquad
\begin{aligned}
\text{maximise} \quad & \sum_{r\in R}\sum_{i\in\tilde{I}_r} \langle w_r, x_r^i\rangle \lambda_r^i \\
\text{subject to} \quad & \sum_{r\in R}\sum_{i\in\tilde{I}_r} C_{\mathcal{E},r} x_r^i \lambda_r^i = b_{\mathcal{E}}, \\
& \sum_{r\in R}\sum_{i\in\tilde{I}_r} C_{\mathcal{I},r} x_r^i \lambda_r^i \leq b_{\mathcal{I}}, \\
& \sum_{i\in\tilde{I}_r} \lambda_r^i = 1, && r \in R, \\
& \lambda_r^i \geq 0, && r \in R, i \in \tilde{I}_r, \\
& \lambda_r^i = 0, && r \in R, i \notin \tilde{I}_r.
\end{aligned}
$$

This problem is often called the (linear relaxation of the) *Restricted Master Problem*. Let us assume that this problem is feasible (otherwise we would have to solve an auxiliary problem to construct a feasible solution first) and denote by

$$\tilde{x}_r = \sum_{i\in\tilde{I}_r} \lambda_r^i x_r^i, \qquad r \in R,$$

an optimal solution. This solution is not necessarily optimal for the original problem, because the set of feasible solutions of the restricted problem is smaller. In order to improve this solution (or to prove that $\tilde{x}$ is indeed optimal) one uses an approach motivated by the simplex method. Let $\tilde{y} = (\tilde{y}_{\mathcal{E}}, \tilde{y}_{\mathcal{I}}, z) \in \mathbb{R}^{\mathcal{E}} \times \mathbb{R}^{\mathcal{I}}_+ \times \mathbb{R}$ be an optimal solution of the dual linear program of (RMP) (one multiplier for each equality and inequality constraint). Then one searches a variable λ_r^i, $r \in R$, $i \in \{1, \ldots, \bar{n}_r\}$, with positive reduced costs

$$\langle w_r, x_r^i\rangle - \langle C_{\bullet,r}^T \tilde{y}, x_r^i\rangle - z > 0.$$

It can be worked out that the search for a variable with maximum reduced costs corresponds to solving the following independent subproblems for each $r \in R$

$$
\begin{aligned}
\text{maximise} \quad & \langle w_r + c_r, x_r\rangle \\
\text{subject to} \quad & x_r \in \mathcal{X}_r,
\end{aligned}
\qquad (CG - Sub(c_r))
$$

with $c_r = -C_{\bullet,r}^T \tilde{y}$. If the solution of one of these subproblems has positive reduced costs, say $x_{\hat{r}}^{\hat{i}}$, then the restriction $\lambda_{\hat{r}}^{\hat{i}} = 0$ is removed from (RMP), which effectively means adding one column to the master problem.

In order to make column generation applicable in practice, several improvements are required [27]. For example, one can add more than one column with positive reduced costs to the master problem or remove unnecessary old ones. Throughout this work we will not deal with column generation approaches in the first place but focus on Lagrangian relaxation. The important fact about column generation, and the reason

why we mentioned it at all, is the following simple observation: The column generation subproblem ($CG - Sub(c_r)$) and the subproblems appearing in Lagrangian relaxation ($Sub^r(c_r)$) are identical. This means that some of the techniques developed in this work (in particular the *Dynamic Graph Generation* approach presented in Chapter 4) cannot only be applied in Lagrangian relaxation but also in column generation approaches.

3.5 Solving the Lagrangian Relaxation: First Order Methods

In order to obtain (upper) bounds as well as approximate (fractional) solutions of (CP) one approach is to solve the Lagrangian relaxation (LD). These problems are typically solved using *first order methods* like *subgradient methods* (see, *e. g.*, Nesterov [78]) or *bundle methods* (see, *e. g.*, Hiriart-Urruty and Lemaréchal [55]). We will sketch this general approach in this section.

The problem (LD) may be written as

$$\begin{aligned} \text{minimise} \quad & f(y) \\ \text{subject to} \quad & y \in \mathbb{R}^{\mathcal{E}} \times \mathbb{R}^{\mathcal{I}}_{+}, \end{aligned}$$

where

$$\begin{aligned} f(y) &= \langle b, y \rangle + \sum_{r \in R} f_r(y), \\ f_r(y) &= \max \left\{ h_r(x_r) - \langle y, C_{\bullet,r} x_r \rangle \colon x_r \in \mathcal{X}_r \right\}, \qquad r \in R. \end{aligned}$$

The functions f_r are convex, because they are the maximum over a compact set of affine functions. In particular, each point $x_r \in \mathcal{X}_r$ induces the affine function

$$y \mapsto h_r(x_r) - \langle y, C_{\bullet,r} x_r \rangle.$$

This implies that f is also a convex function as sum of convex functions. Furthermore each point $x = (x_r)_{r \in R} \in \mathcal{X} = \bigtimes_{r \in R} \mathcal{X}_r$ defines an affine minorant of f

$$\hat{f}_x(y) = \sum_{r \in R} h_r(x_r) + \langle g(x), y \rangle, \tag{3.1}$$

where

$$g(x) = b - \sum_{r \in R} C_{\bullet,r} x_r. \tag{3.2}$$

Note that the function f is non-smooth in general (*e. g.*, it is piecewise affine if $\mathcal{X}$ is finite). Hence (LD) is a convex, non-linear, non-smooth optimisation problem.

A standard algorithmic approach to solve these problems are first order methods (Bonnans et al. [6, Chapter 8], Hiriart-Urruty and Lemaréchal [55]). A first order method requires that the objective function, say f as in our case, is given by a black box oracle that computes for any given y

- the function value $f(y)$ and
- a subgradient $g \in \partial f(y)$ of f in y.

The subdifferential $\partial f(y)$ of a function f at the point y is defined as follows (Hiriart-Urruty and Lemaréchal [55, Chapter VI]).

Definition 3.2 Let $Y \subseteq \mathbb{R}^m$, convex, $f\colon Y \to \mathbb{R}$ be a convex function, $y \in Y$. The *subdifferential* of f in y is the set

$\partial f(y)$

$$\partial f(y) := \{s \in \mathbb{R}^m \colon \forall\, y' \in Y, f(y') \geq f(y) + \langle s, y' - y\rangle\}.$$

The elements $s \in \partial f(y)$ are the *subgradients* of f at y.

So each subgradient $s \in \partial f(y)$ defines an affine minorant

$$y' \mapsto f(y) + \langle s, y' - y\rangle$$

that minorises f.

The basic idea behind first order methods is to iteratively generate a sequence of points $(y_k)_{k\in\mathbb{N}}$ by evaluating the function at certain candidate points $(\bar{y}_k)_{k\in\mathbb{N}}$ be means of the black box oracle. This evaluation returns the function value $f(\bar{y}_k)$ along with a subgradient $g_k \in \partial f(\bar{y}_k)$. The next iterate is then found by determining a direction $d_k \in \operatorname{span}\{g_i \colon i \in \{1, \ldots, k\}\}$ and a step size $\alpha_k \geq 0$ and $y_{k+1} = y_k + \alpha_k d_k$. The concrete rules how the candidate points, the directions and step sizes are computed differ among the first order methods.

In order to apply a first order method we must provide a black box oracle for the actual dual function f. The computation of $f(y)$ requires the values $f_r(y)$ for all $r \in R$, so we must solve the subproblems $(Sub^r(c_r))$, $r \in R$, for $c^r = -C^T_{\bullet,r} y \in \mathbb{R}^{n_r}$. Note that we assumed in this section that these subproblems are easy to solve. Let $x^* = (x^*_r)_{r\in R} \in \mathcal{X}$ be a vector of optimal solutions of these subproblems, *i. e.*

$$x^*_r(y) \in \operatorname{Argmax}\{h_r(x_r) - y^T C_{\bullet,r} x_r \colon x_r \in \mathcal{X}_r\}.$$

By (3.1) and (3.2) this defines an affine minorant

$$\hat{f}_{x^*}(y') = f(y) + \langle g(x^*), y' - y\rangle \leq f(y') \quad \text{for all } y' \in Y$$

and $\hat{f}_{x^*}(y) = f(y)$, so $g(x^*) \in \partial f(y)$ is a subgradient (note that *each* optimal solution x^* defines a subgradient, which is not unique in general). In other words, the optimal solutions of the subproblems $(Sub^r(c_r))$, $r \in R$, give us the function value $f(y)$ as well as a subgradient $g(x^*)$ in y. This provides a black box oracle and hence is all we need to apply a first order method to the Lagrangian relaxation (LD).

3.5.1 Cutting Planes

Quite often the number of potential constraints in an integer program is very large in practice. Adding all constraints to the model right from the beginning often leads to huge memory requirements and algorithmic difficulties. A classical example is the standard, cut-based formulation of the *Travelling Salesman Problem* (Dantzig et al. [25]), which contains an exponential number of constraints. From the theoretical point of view, the *ellipsoid method* (see, *e. g.*, Schrijver [87]) can be used to solve the linear relaxation of problems even in the presence of exponentially many constraints, given that they can be separated efficiently (*i. e.*, in polynomial time). In practice, however, the (dual) *simplex method* proved to work well with cutting plane approaches, although its running time is not polynomial in theory.

In the case of the TTP model (TTP-clq) in Section 2.5, the number of constraints is very large, as well. Thus, in order to be efficient in practice, we have to separate the clique inequalities (2.7). In our implementation we separated all maximal cliques induced by the runs for each two trains (note that because of the behaviour dependent running times and headway times even the number of maximal cliques on two trains may become quite large).

Our main solution approach employs Lagrangian relaxation to be solved by a first order method, in particular by a bundle method. We refer to Chapter 5 for a description of this method. Fortunately, cutting planes can be combined with this bundle method, see Section 5.2.1.

3.6 Application to the TTP

In this section we investigate how the TTP (see Chapter 2) fits into the framework sketched in this chapter.

In the previous chapter we formulated two slightly different models for the TTP. Both models are based on time expanded networks for the train graphs with coupling inequality constraints for the capacity restrictions in the stations. They differ in the way the headway restrictions are modelled. The first model (TTP-clq) in Section 2.5 used additional so called clique inequalities to forbid the simultaneous use of train runs that have a headway conflict. The second model (TTP-cfg) in Section 2.6 modelled the feasible use patterns of conflict-free configurations in additional configuration networks, which are linked to the train graphs via simple equality constraints. Both models are problems of type (CP). The ground sets $\mathcal{X}_r$, $r \in R$, correspond to the set of feasible paths $\mathcal{P}_p$, $p \in R \cup A^I$, in the train graphs respective configuration networks. The capacity constraints as well as the clique inequalities form linear inequality constraints, the coupling configuration constraints form linear equality constraints. The only difference is the direction of optimisation: the TTP-models *minimise* their objective whereas (CP) *maximises* the objective, but this difference is merely a multiplication of the objective functions by -1.

The main solution approach we want to employ is Lagrangian relaxation. In this case

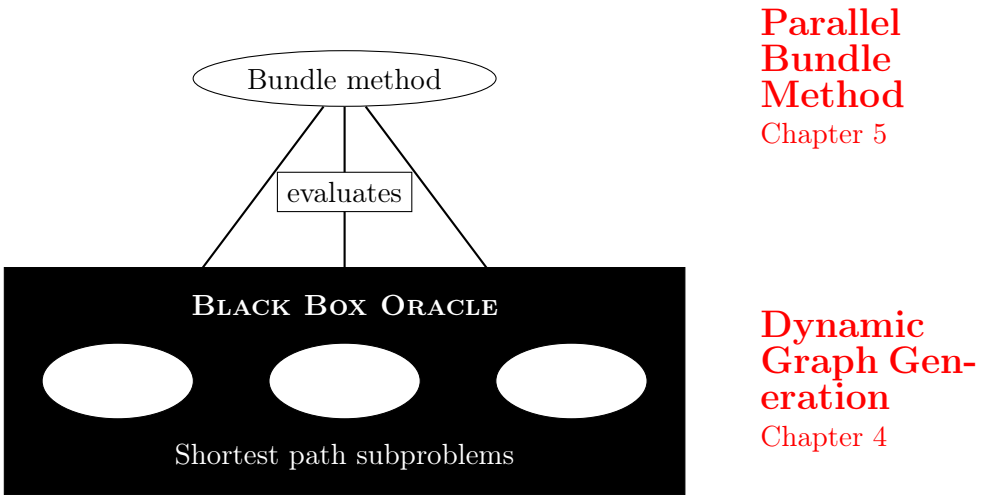


Figure 3.1: Structure of solution method. The basic approach is to use a bundle method to solve the Lagrangian relaxation. The bundle method uses the black box oracle to evaluate the dual function $f(y)$ at certain points. In order to improve the efficiency of the solution process, we developed two advanced techniques: First, *Dynamic Graph Generation*, presented in Chapter 4, improves the solution of subproblems by reducing the size of the networks to be kept in memory drastically. Second, the *Parallel Bundle Method* dynamically selects several subspaces in parallel that correspond to strongly violated coupling constraints.

the subproblems ($Sub^r(c_r)$) read

$$\begin{aligned} \text{maximise} \quad & \langle w_r + c_r, x_r \rangle \\ \text{subject to} \quad & x_r \in \mathcal{P}_r, \end{aligned}$$

with $r \in R$ where R denotes the set of all trains (train graphs) plus possibly the set of all configuration networks, depending on the model. Note that these subproblems are shortest path problems in the respective networks, thus easy to solve. This means, by the considerations in Section 3.5, that we can apply first order methods to solve the Lagrangian relaxation.

The first order method that we used in our application is a proximal bundle method (see Chapter 5 for an introduction of bundle methods). In the project it quickly turned out that the models grow rapidly and are very hard to solve for real world instances (see Chapter 6 for some numerical tests). We developed two techniques that should help to improve the solution process. The first one, Dynamic Graph Generation presented in Chapter 4, aims at reducing the size of the networks to be kept in memory. Without sacrificing accuracy only small parts of the networks have to be stored, which greatly improves memory usage and also the performance of the subproblem evaluations. The second technique is a Parallel Bundle Method to be described in Chapter 5. There we do not deal with the subproblems but with the optimisation algorithm itself. In order

to improve the solution process, the parallel bundle method selects dynamically and asynchronously several disjoint subspaces of dual multipliers, so that these subspaces correspond to strongly violated coupling constraints. The idea is that the algorithm focuses first on heavily violated constraints, which requires only the solution of those subproblems that interact with these constraints, possibly reducing the computational effort. A sketch of the structure of our solution approach is shown in Figure 3.1.

4 Dynamic Graph Generation

4.1 Introduction

Time expanded networks have proved to be a very useful modelling tool for many discrete optimisation problems, the TTP introduced in Chapter 2 being one example. One of the major drawbacks of this approach is the rapidly increasing model size when the number of time steps increases, because a larger time horizon has to be covered or a finer discretisation step size is needed. This increasing model size leads to huge memory requirements as well as to increased running times of the algorithms working on those models. (Back in the old days—by computer science standards, *i. e.* five to ten years ago—when most computers were 32 bit machines and had a natural limit of about 4 GB addressable main memory large memory requirements were much more threatening than today. But increasing computing power leads to more ambitious requests from practice, so saving resources should never be out of fashion).

In this chapter we consider a special class of models involving large time expanded networks, which can be stated as follows.

Problem

$$\begin{aligned} \text{minimise} \quad & \sum_{r \in R} h_r(x_r) = \sum_{r \in R} \langle w_r, x_r \rangle \\ \text{subject to} \quad & x_r \in \mathcal{P}_r, \qquad r \in R, \\ & \sum_{r \in R} C_{\bullet,r} x_r \leq b. \end{aligned}$$

Here the sets $\mathcal{P}_r$, $r \in R$, represent sets of (characteristic vectors of) feasible paths in time expanded networks $G_r^T = (V_r^T, A_r^T)$, $r \in R$, according to Definition 2.2. The objective function is linear and sums up the weighted cost terms for each arc. For a path $P \in \mathcal{P}_r$ we will frequently use the notation

$$h_r(P) = \sum_{a \in A_r^T(P)} w_{r,a} = \langle w_r, \chi_P \rangle,$$

where $w_r = (w_{r,a})_{a \in A_r^T}$ and χ_P denotes the characteristic vector of P.

This is exactly the kind of model we use for the TTP introduced in Chapter 2. In Chapter 3 we described general solution approaches for these models, namely Lagrangian

relaxation (see Section 3.3) and column generation (see Section 3.4). We showed that both approaches require the repeated solution of subproblems for each $r \in R$ ($(Sub^r(c_r))$ resp. $(CG - Sub(c_r))$) with the following structure, which will be the problem to be dealt with in this chapter (up to the sense of the objective, in this chapter we always consider minimisation):

Problem

$$\begin{aligned} \text{minimise} \quad & h_r(x_r) + \langle c_r, x_r \rangle \\ \text{subject to} \quad & x_r \in \mathcal{P}_r, \end{aligned}$$

where $c_r \in \mathbb{R}^{n_r}$ defines a linear augmented cost term possibly depending on Lagrange multipliers.

We assume that the sets $\mathcal{P}_r$, $r \in R$, are structured so that the solution of one of these subproblems corresponds to a shortest path computation in the corresponding time expanded graph G_r^T.

The Dynamic Graph Generation technique to be developed in this chapter focuses on the solution of these subproblems. The main difficulty, as mentioned before, is the enormous size of the time expanded networks. In principle, any path in the network can be the shortest path depending on the augmented cost term c_r, and if that term changes the shortest path might change as well. Therefore, without further efforts, we would have to store the whole network, because each arc may be contained in the shortest path at any time.

However, in practice the situation is usually different. First, the basic objective function h_r, $r \in R$, often has a nice, well defined structure. In many scheduling applications the objective is to minimise the overall completion time implying that "early" paths are cheaper than "late" paths. For example, classical objective functions are the total (sum of) completion time and sum of lateness, see, *e. g.*, Gawiejnowicz [39], Graham et al. [45]. They are also used in TTP problems, *e. g.*, Brännlund et al. [11] aimed to minimise the accumulated delay. The situation is similar in our case (see Section 2.7): Passenger trains should arrive at each station with as little delay as possible and freight trains should arrive at their final destination as early as possible.

The second observation is that the augmented cost term does not change arbitrarily but is adapted by the underlying optimisation algorithm (the first order method solving the Lagrangian dual or the master problem in the case of column generation). Quite often, between two successive solutions of the subproblems, the new augmented cost term is close to the previous term in the sense that the costs of most arcs change only slightly. Therefore one might expect that the optimal solution of the subproblem also changes only slightly.

These two ingredients motivate the following algorithmic approach. Instead of keeping the whole time expanded network in memory, we only store that part of the network

that has been interesting in previous iterations (*i. e.*, that contains all paths returned in previous subproblem evaluations and maybe a little bit more). These are often the parts of the network that contain the early time steps. When the next subproblem should be solved, the augmented cost term is relatively close to the previous one. Because the basic cost function h is well structured (prefers early paths), we can reasonably expect that the new shortest path is also contained in these early time steps or is at least close to them. Instead of solving the shortest path problem on the whole network, we solve the shortest path problem only on that small part. We will see that it is either possible to verify that the solution of this restricted shortest path problem is also a shortest path in the whole network, or we get information how the subnetwork should be extended. In the latter case the subnetwork that we have to keep in memory must be increased by dynamically adding new nodes and arcs to the network. We will show how this approach can be used to develop an algorithm, that solves the shortest path problems *exactly*, *i. e.* without any loss of information, but at the same time reduces the size of the subnetwork to be kept in memory dramatically. This will even enable us to deal with the case that the theoretic time horizon of the time expanded networks is infinite (under some reasonable assumptions).

Time expanded networks also appear in the literature in the context of *flows over time* or *dynamic flows*, see Aronson [2], Kotnyek [61], Skutella [90] and the references therein for an overview on the topic and Helmberg and Röhl [53], Kamiyama et al. [57] for some applications. The shortest path subproblems are among the best investigated combinatorial optimisation problems. The most famous algorithm, which is still the basis of many modern algorithms, has been presented by Dijkstra [30]. A typical application for shortest paths, and arguably the most important one in the last decades, are fast point-to-point shortest paths in large networks, *e. g.* road networks Goldberg et al. [41], Potamias et al. [81]. A survey over point-to-point problems is Goldberg [40], an extensive computational study can be found in Klunder and Post [60]. These algorithms typically require some preprocessing and are designed to solve quickly shortest path problems with different source and sink nodes in the same graph, possibly only approximately. In contrast, the cost function in our problems may change arbitrarily (*i. e.*, the weights may increase and decrease) at every iteration as long as the initial cost structure satisfies a reasonable technical condition. Furthermore, we need to solve them exactly. Shortest path problems have also been considered in the dynamic case, in which the transit-time of an arc depends on the entering time of the path, see, *e. g.*, Ahuja et al. [1], Delling and Wagner [26]. Usually these algorithms are designed for modelling dynamic scenarios like traffic jams in large, otherwise static road networks, and work under similar assumptions as in the static case. Many of the aforementioned shortest path algorithms (*e. g.*, [41]) base on A^* approaches, see Hart et al. [48]. These algorithms use a heuristic, which is a lower bound on the distance from each node to the destination, to guide the Dijkstra algorithm more directly to the destination. The special situation of solving shortest path subproblems in Lagrangian relaxation has been investigated by Yanagisawa [94] who showed that an A^* based search algorithm may outperform a simple Dijkstra algorithm when used to solve the Lagrangian dual problem of a multi-commodity flow problem. This approach requires that all coupling constraints are inequalities that can only increase the arc costs via the

Lagrange multipliers. In contrast, our algorithm can handle negative augmenting costs as well when combined with separation of those constraints. Nevertheless, by exploiting the structure of the initial cost function, their approach may be used to solve the shortest path problems on the time expanded networks without the requirement to store nodes on expensive paths which may lead to a stored subgraph of similar size compared to the approach presented in Section 4.3.

This chapter is organised as follows. In Section 4.2 we state the basic algorithmic framework for the solution of the shortest path problems and discuss the finiteness of this approach in models with infinite time horizon. We also specify the required property for the subnetwork to be kept in memory. The construction of these subnetworks will be the main work. In Section 4.3 we give a first simple construction of such networks. This construction may be further improved in certain cases, so it is extended to our final algorithm in Section 4.4. We summarise the chapter in Section 4.4.5 and finally we discuss in Section 4.5 how dynamic graph generation can be applied in the context of our practical TTP application.

The results presented in this chapter are taken from [35], partially verbatim. The main differences are the detailed discussion of the infinite time horizon (Section 4.2.1) and the relation to our TTP models in Section 4.5. The notation has been adapted so as to match the notation used in this thesis.

4.2 Dynamic Graph Generation

We start this section by introducing some notation to be used throughout this chapter. According to the structure of the objective function described in Section 4.1 we consider objective functions $h\colon A^T \to \mathbb{R}$ and write

$$h(P) = \sum_{a \in A^T(P)} h(a)$$

where $P \in \mathcal{P}$ is a feasible path in G^T.

Let P be a path in the time expansion of a graph $G = (V, A)$, *i. e.* $P \subset G^T = (V^T, A^T)$ with

$$P = (u_1, t_{u_1}) \dots (u_1, t'_{u_1})(u_2, t_{u_2}) \dots (u_k, t'_{u_k}).$$

Then we denote by $P_{|G}$ the corresponding path in G, *i. e.*

$P_{|G}$

$$P_{|G} := u_1 u_2 \dots u_k.$$

Throughout this chapter we will assume $T = \mathbb{N}$ for the set of time steps, *i. e.*, it is infinite.

Dynamic graph generation focuses on the repeated solving of the oracle subproblem ($Sub^r(c_r)$) in the case of a Lagrangian relaxation approach or the pricing subproblem ($CG-Sub(c_r)$) in the case of a column generation approach for some fixed $r \in R$. The goal is to exploit the fact that the objective functions in subsequent subproblem evaluations differ in some controllable way because of how the solution algorithm (*e. g.* a subgradient or bundle method) works. Because the single subproblems are all independent and of

equivalent type, we will drop the subscript r throughout the chapter for the ease of notation.

We start by stating again the kind of subproblem that we need to solve repeatedly. Let $G = (V, A)$ be a directed, acyclic network (except for loops), with a unique source node $\hat{u}$ and a unique sink node $\check{u}$ ($\check{u}$ must not have a loop), $T = \mathbb{N}$ the set of time indexes and $d\colon A \to 2^{\mathbb{N}}$ a traversal time function. Then we denote by $G^T = (V^T, A^T)$ the time expansion of G according to Definition 2.2 and by $\mathcal{P}$ the set of all $(\hat{u}, 1)$-$(\check{u}, t_{\check{u}})$ paths in G^T with $t_{\check{u}} \in T$. We assume further that the base graph G has the following additional structural properties.

The node set V is partitioned into a set $[V]$ of *clone nodes*. For each node $u \in V$ we denote the unique clone node that contains u by $[u] \in [V]$. All nodes contained in a single clone node are closely related and considered "similar" in the following sense. Let $[w] \in [V]$, then no two nodes $u, v \in [w]$ contained in the same clone node are adjacent and u and v have exactly the same neighbourhood in G (disregarding loops). Formally these conditions read for $u, v, w \in V$ [V]

(K1) $u, v \in [w], u \neq v \Rightarrow (u, v) \notin A$ (note, loops are allowed inside a class),

(K2) $(u, v) \in A \Rightarrow [u] \times [v] \subseteq A$,

(K3) $[\hat{u}] = \{\hat{u}\}, [\check{u}] = \{\check{u}\}$.

The set of nodes V is partitioned into the clone nodes $[V]$. Based on this partition we define the *clone graph* $[G] = ([V], [A])$ with

$$[A] := \{([u], [v])\colon (u, v) \in A, u \neq v\}.$$

Each walk $P \subseteq A$ canonically induces a path in $[G]$. In more detail, if $P = u_1 \dots u_k$, then $[P] = [u_{i_1}] \dots [u_{i_l}]$, $1 = i_1 < \dots < i_l = k$, refers to the path of clone nodes visited in sequence by P with duplicates removed. Likewise, if P is seen as a subset of arcs of A, then $[P]$ refers to the induced arc set of $[A]$. If

$$P = (u_1, t_{u_1}) \dots (u_1, t'_{u_1})(u_2, t_{u_2}) \dots (u_k, t'_{u_k})$$

is a path in G^T, then $[P]$ denotes the induced path in the clone graph, *i. e.* [P]

$$[P] = [P_{|G}] = [u] \dots [v]$$

and is called the *route* of P.

We will often refer to some finite subgraph $\tilde{G}^T = (\tilde{V}^T, \tilde{A}^T) \subset G^T$, $|\tilde{V}^T|, |\tilde{A}^T| < \infty$ of G^T. For such a finite subgraph we will frequently need to include all nodes and arcs at earlier time steps, the *closure* of that graph.

Definition 4.1 Let $\tilde{V}^T \subset V^T$ be a subset of the nodes of G^T. The *closure* of $\tilde{V}^T$ is the graph

$$\operatorname{cl} \tilde{V}^T = (\tilde{V}^{\mathrm{cl}}, \tilde{A}^{\mathrm{cl}})$$ $\operatorname{cl} \tilde{V}^T = (\tilde{V}^{\mathrm{cl}}, \tilde{A}^{\mathrm{cl}})$

with

$$\tilde{V}^{\mathrm{cl}} := \left\{(v,t) \in V^T \colon (v',t') \in \tilde{V}_T, [v'] = [v], t \le t'\right\} \cup (V \times \{1\}),$$
$$\tilde{A}^{\mathrm{cl}} := (\tilde{V}^{\mathrm{cl}} \times \tilde{V}^{\mathrm{cl}}) \cap A^T.$$

The *closure of a subgraph* $\tilde{G}^T = (\tilde{V}^T, \tilde{A}^T)$ is the closure of its node set

$$\mathrm{cl}\,\tilde{G}^T := \mathrm{cl}\,V^T(\tilde{G}^T) = \mathrm{cl}\,\tilde{V}^T.$$

Hence the closure $\mathrm{cl}\,\tilde{V}^T$ contains all nodes $(u,t) \in V^T$ for which $\tilde{V}^T$ contains an equivalent node $v \in [u]$ at the same or at a later time step together with the induced arcs.

Later we will exploit relations between the objective function and the structure of G^T resp. G. These relations will be expressed in terms of $[G]$.

Remark 4.2 A clone partition of an acyclic directed graph with a single source node and a single sink node is quickly determined. Indeed, ignoring loops, call two nodes *twins* if they have an identical set of predecessors and an identical set of successors. This declares an equivalence relation on the nodes that can be checked efficiently. Its equivalence classes satisfy (K1) because there cannot be an arc between twins, (K2) because the adjacency relation is the same for all twins of an equivalence class, and (K3) because we required a single source and a single sink node. Later, however, we will require some conditions on the objective function c w. r. t. the clone partition (condition (C) in Section 4.3) that seem natural for typical clone partitions arising in practical applications but that might not always be satisfied for this generic partition.

Remark 4.3 Clone partitions are a convenient abstraction motivated by our practical application, the TTP. Indeed, we have already seen an example of a graph G together with its clone graph $[G]$ in Section 2.3, Figure 2.2. The run node (u, Run) and the wait node (u, Stop) belong to the same station and form one clone node

$$[(u,\mathit{Run})] = [(u,\mathit{Stop})] = \{(u,\mathit{Run}),(u,\mathit{Stop})\} \simeq u$$

and this partition fulfils (K1)–(K3), possibly after adding artificial source and sink nodes. In fact, the clone graph corresponds to the *route graph* of a train (denoted by $\bar{G}_r$ in Chapter 2).

Note that the framework in this chapter is more general than in the case of the TTP in Chapter 2. There we assumed that the clone graph (*i. e.* the route graph) is a path. In our current framework we allow more general structures leading to more general clone graphs, see Figure 4.1 for an example. This might, *e. g.*, be used to extend the TTP model by certain routing aspects by providing several possible train routes to choose from.

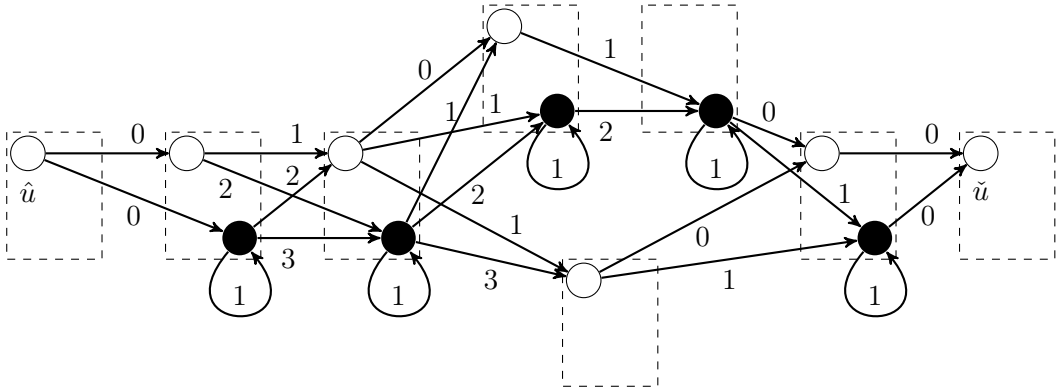


Figure 4.1: Base graph $G = (V, A)$ with alternative routes. The dashed boxes are independent clones $[u]$. Each node contained in $[u]$ can be thought of some state of operation. The white nodes are modes in which waiting is not allowed, the black nodes are modes in which waiting is allowed. The numbers are the traversal times along the arcs, *i. e.*, in this example all sets $d((u, v))$, $uv \in A$, contain exactly one element.

Remark 4.4 Because the graph G is acyclic (except for loops), the graphs $[G]$ and G^T are acyclic, too. This leads to the nice property that we have a well defined topological partial order on the nodes V (resp. $[V]$ or V^T), say $\leq_G$, defined by

$$u \leq_G v \iff \exists \text{ path } P = uPv \subseteq G$$

for all nodes $u, v \in V$. We will later exploit this property algorithmically and also in proofs, *e. g.*, by using induction in topological or reverse topological order of the nodes.

The task is to solve a shortest path subproblem w. r. t. an augmented cost function $h^{\text{aug}} \colon A^T \to \mathbb{R}$, coming from the Lagrange multipliers in Lagrangian relaxation or the reduced costs in column generation, respectively.

Problem (Full Subproblem) Given an *augmented cost function* $h^{\text{aug}} \colon A^T \to \mathbb{R}$ determine an optimal path $\hat{P}(h^{\text{aug}}) \in \mathcal{P}$ of

$$(\text{DyGG}(h^{\text{aug}})) \qquad \begin{aligned} \text{minimise} \quad & h^{\text{aug}}(P) \\ \text{subject to} \quad & P \in \mathcal{P}. \end{aligned}$$

In general, the augmented cost function h^{aug} changes each time the subproblem should be solved. The difficulty in solving this problem comes from the fact that the graph G^T

can be very large (if T is large but finite) or even infinite (if T is infinite) and thus the set of all possible solutions $\mathcal{P}$ can also be very large or even infinite.

During the solution process, however, only a small part of the network turns out to be "interesting" in the sense that this part has been important in the previous iterations. In particular, let $\mathcal{P}^{\text{prev}}$ denote the set of the computed optimal solutions (*i. e.* shortest paths) of the previous iterations and $G^{\text{act}} = G[\mathcal{P}^{\text{prev}}]$ the graph induced by these paths. The set of all paths $\mathcal{P}$ that are contained in G^{act} is denoted by $\mathcal{P}^{\text{act}}$. Note, $\mathcal{P}^{\text{prev}} \subseteq \mathcal{P}^{\text{act}}$ but in general $\mathcal{P}^{\text{prev}} \neq \mathcal{P}^{\text{act}}$. G^{act} is large enough so that solving the shortest path problem with the objective function of some earlier iteration restricted to G^{act} would return a correct shortest path in G^T (because by definition G^{act} contains at least one such shortest path). It is reasonable to expect that this subgraph is important for the next shortest path computation, too, and may even already contain a shortest path on G^T w. r. t. h^{aug}, but certainly this property does not necessarily hold in general. The basic idea is now to solve the shortest path problem on a small finite subgraph $G^{\text{cur}} = (V^{\text{cur}}, A^{\text{cur}})$ of G^T, the *current subgraph* that is currently stored in memory. The current subgraph will always contain G^{act}, *i. e.* $G^{\text{act}} \subseteq G^{\text{cur}} \subset G^T$. However, G^{cur} is not an arbitrary graph that contains G^{act} but is required to contain additional arcs so that G^{cur} has some important structural properties (see below). If G^{cur} possesses these properties then it will be called a *valid subnetwork*. Indeed, we will solve the shortest path problem only in G^{cur} w. r. t. the augmented cost function h^{aug} on A^{act} and the original cost function h on $A^{\text{cur}} \setminus A^{\text{act}}$. The additional arcs of G^{cur} compared with G^{act} are used to detect whether the shortest path on G^{cur} is indeed a shortest path on G^T. More precisely, we say G^{cur} is valid if it fulfils the following condition.

Definition 4.5 Let $G^{\text{act}} = (V^{\text{act}}, A^{\text{act}}) \subset G^T$ be an active subnetwork of G^T and let $G^{\text{cur}} = (V^{\text{cur}}, A^{\text{cur}})$ with $G^{\text{act}} \subset G^{\text{cur}} \subset G^T$ with $|V^{\text{cur}}| < \infty$ be a finite subnetwork containing G^{act}. Define $h^{\text{cur}} \colon A^T \to \mathbb{R}$ as

$$h^{\text{cur}}(a) = \begin{cases} h^{\text{aug}}(a), & \text{if } a \in A^{\text{act}}, \\ h(a), & \text{otherwise.} \end{cases} \tag{4.1}$$

Let $\mathcal{P}^{\text{cur}} = \{P \in \mathcal{P} \colon P \subseteq G^{\text{cur}}\}$ be the set of feasible paths in G^{cur}. Then G^{cur} is a *valid subnetwork* if it holds

(V)
$$P^* \in \operatorname{Argmin}\{h^{\text{cur}}(P) \colon P \in \mathcal{P}^{\text{cur}}\}, P^* \subset G^{\text{act}} \Rightarrow P^* \in \operatorname{Argmin}\{h^{\text{aug}}(P) \colon P \in \mathcal{P}\},$$

i. e., if each shortest path w. r. t. h^{cur} in G^{cur} that is contained in G^{act} is also a shortest path on G^T w. r. t. h^{aug}.

The following two simple properties of valid subnetworks follow directly from the definition.

Observation 4.6 *Let G^{act} be an active subnetwork. Then*

(i) $G^{cur} := G^{act}$ is a valid subnetwork if and only if there is a path $P^* \in \mathcal{P}^{act}$ so that $P^* \in \operatorname{Argmin}\{h^{aug}(P) \colon P \in \mathcal{P}\}$,

(ii) if $\operatorname{Argmin}\{h^{cur}(P) \colon P \in \mathcal{P}^{cur}\} \cap \mathcal{P}^{act} = \emptyset$ then G^{cur} is a valid subnetwork.

In other words, G^{act} is itself a valid subnetwork if and only if it contains a shortest path in G^T w. r. t. h^{aug}, and if each shortest path in G^{cur} w. r. t. h^{cur} is not *contained in G^{act}, then G^{cur} is also a valid subnetwork.*

Proof. Directly from the definition. □

Now suppose that, given an active subgraph G^{act}, we are able to compute a valid current subgraph G^{cur}. Denote the current subproblem on G^{cur} by

Problem (Current Subproblem) Given an *active subgraph* G^{act}, an *augmented cost function* h^{aug} and a corresponding valid subnetwork $G^{\text{cur}} \supset G^{\text{act}}$, determine an optimal path $P^*(G^{\text{cur}}, h^{\text{aug}}) \in \mathcal{P}^{\text{cur}}$ of

$$(DyGG^{\text{cur}}(G^{\text{cur}},h^{\text{aug}})) \qquad \begin{aligned} \text{minimise} \quad & h^{\text{cur}}(P) \\ \text{subject to} \quad & P \in \mathcal{P}^{\text{cur}}, \end{aligned}$$

with h^{cur} defined as in (4.1).

Using this we are able to solve the shortest path problem (DyGG(h^{aug})) on G^T using the schematic algorithm shown in Algorithm 4.1. Note that the input data always has a finite representation ($G^T = (V^T, A^T)$ is given by $G = (V, A)$ and traversal time d, h^{aug} is given as oracle, $\mathcal{P}^{\text{prev}}$ is finite).

Algorithm 4.1: Dynamic Shortest Path (scheme)

Input : Graph $G^T = (V^T, A^T)$, cost function $h^{\text{aug}} \colon A^T \to \mathbb{R}$, finite $\mathcal{P}^{\text{prev}} \subset \mathcal{P}$
Output : A shortest path $P \subseteq G^T$ w. r. t. h^{aug}
repeat
  determine valid G^{cur} based on $G^{\text{act}} = G^T[\mathcal{P}^{\text{prev}}]$
  compute $P^* \in \operatorname{Argmin}\{h^{\text{cur}}(P) \colon P \in \mathcal{P}^{\text{cur}}\}$
  set $\mathcal{P}^{\text{prev}} \to \mathcal{P}^{\text{prev}} \cup \{P^*\}$
until $P^* \subset G^{act}$
return P^*

Because G^{cur} is *valid* the path returned by this algorithm is indeed a solution of the shortest path problem in G^T w. r. t. h^{aug}. In each iteration of this algorithm a shortest path is computed. Afterwards it is checked whether this path can be guaranteed to be a global shortest path in G^T w. r. t. h^{aug} using the validity of G^{cur}. If this is the case the path is returned, otherwise the returned path is added to the active subgraph. This approach enlarges G^{act} and (usually) also the corresponding current subgraph G^{cur}. Note that we only need to store the subgraph G^{cur} in memory instead of the full graph G^T in order to run the algorithm. Because this subgraph G^{cur} is enlarged dynamically during the run of the algorithm if required, we call this approach *Dynamic Graph Generation*.

Dynamic Graph Generation

There are two aspects of Algorithm 4.1 that are unclear up to now. First, a priori it is not clear that Algorithm 4.1 finishes after a finite number of iterations (in fact, it may run forever if G^T contains no shortest path w. r. t. h^{aug} which is possible if G^T is infinite). We will see in Section 4.2.1 that under reasonable assumptions on the cost functions and the algorithm solving ($DyGG^{\text{cur}}(G^{\text{cur}},h^{\text{aug}})$) the algorithm will stop after a finite number of iterations. But the main difficulty is the computation of a *valid* current subgraph G^{cur} for a given G^{act}. Because G^{cur} is the subgraph that we have to keep in memory it is important to keep that subgraph as small as possible. In the remainder of this chapter we will develop an approach to efficiently construct a valid subgraph G^{cur} that is a reasonably small extension of G^{act}.

4.2.1 Finiteness of the Outer Iteration

In this section we will investigate the properties of Algorithm 4.1 w. r. t. an infinite time expanded graph G^T. In order to get the algorithmic scheme Algorithm 4.1 to terminate in finite time we require several conditions on the basic cost function h, the augmented cost function h^{aug} and the oracle computing a solution of the subproblem ($DyGG^{\text{cur}}(G^{\text{cur}},h^{\text{aug}})$). Their main purpose is to guarantee that the problem (DyGG(h^{aug})) has a finite optimal solution and to ensure that this solution can be found and verified after a finite number of iterations. We start by requiring that the solution algorithm of ($DyGG^{\text{cur}}(G^{\text{cur}},h^{\text{aug}})$) behaves nicely.

Definition 4.7 An algorithm solving ($DyGG^{\text{cur}}(G^{\text{cur}},h^{\text{aug}})$) is called *proper* if the returned solution path $P^* = P^*(\check{u}, t_{\check{u}})$ satisfies

$$t_{\check{u}} = \min\{t\colon P = P(\check{u}, t) \in \mathcal{P}^{\text{cur}}, h^{\text{cur}}(P) = h^{\text{cur}}(P^*)\}. \tag{4.2}$$

In other words, the final arrival time of the returned solution must be as small as possible.

Note that any algorithm solving ($DyGG^{\text{cur}}(G^{\text{cur}},h^{\text{aug}})$) can easily be made proper. One only has to modify the algorithm so that the objective value of the arc $a = (u, t_u)(\check{u}, t_{\check{u}}) \in A^T$ arriving at $\check{u}$ is slightly increased to $h^{\text{cur}}(a) + \varepsilon \cdot t_u$ for a sufficiently small $\varepsilon > 0$. So

being proper is no real restriction, but the following example shows the necessity of this requirement.

Example 4.8 Let $G_{L,n} = (V_{L,n}, A_{L,n})$ be the graph given by $G_{L,n}$

$$V_{L,n} = \{\hat{u} = u_1, \dots, u_n = \check{u}\},$$
$$A_{L,n} = \{u_i u_{i+1} \colon i \in \{1, \dots, n-1\}\} \cup \{u_i u_i \colon i \in \{1, \dots, n-1\}\}$$

and

$$d_{L,n} \colon A_{L,n} \to 2^{\mathbb{N}_0}, d_{L,n}(uv) = \begin{cases} \{1\}, & \text{if } u = v, \\ \{0\}, & \text{otherwise.} \end{cases}$$

The time expansion $G^T_{L,n}$ of $G_{L,n}$ is isomorphic to an infinite grid graph, see Figure 4.2.

Now consider the case $n = 2$, the cost functions $h^{\text{aug}} = h = 0$ and assume that the current subgraph is always constructed so that it contains at least one path not contained in G^{act}, *e. g.*

$$G^{\text{cur}} := G_{L,n}[\{(u,t) \colon u \in V, t \leq \bar{t} + 1\}]$$

where

$$\bar{t} := \max\{t \colon (u,t) \in V^{\text{act}}\}.$$

with initial $V^{\text{act}} = \{(u_i, 1) \colon i \in \{1, \dots, n\}\}$. This graph G^{cur} satisfies (V) and is therefore valid. If the algorithm solving $(DyGG^{\text{cur}}(G^{\text{cur}}, h^{\text{aug}}))$ is not proper then it may return the path $P^* = (u_1, 1) \dots (u_1, \bar{t} + 1)(u_2, \bar{t} + 1) \not\subseteq G^{\text{act}}$. Thus Algorithm 4.1 would increase G^{act} by P^* and execute another iteration.

In addition to the algorithm for solving the shortest path problem we need some assumptions on the cost functions h and h^{aug}. We state those assumptions next and discuss their implications afterwards.

(C1) $h^{\text{aug}}(a) \geq h(a)$ for all $a \in A^T \setminus A^{\text{act}}$,

(C2) for each path $P = (u, t_u) \dots (\check{u}, t_{\check{u}})$ in G^T there is a time threshold $\bar{t}(P) \in T$ so that

$$\forall\, P' = (u, t_u) \dots (\check{u}, t'_{\check{u}}), P_{|G} = P'_{|G}, t'_{\check{u}} \geq \bar{t}(P) \text{ it holds } h(P') \geq h(P),$$

(C3) $|\{a \in A^T \colon h^{\text{aug}}(a) \neq h(a)\}| < \infty$,

First we demonstrate that no valid subnetwork might exist if (C1) does not hold.

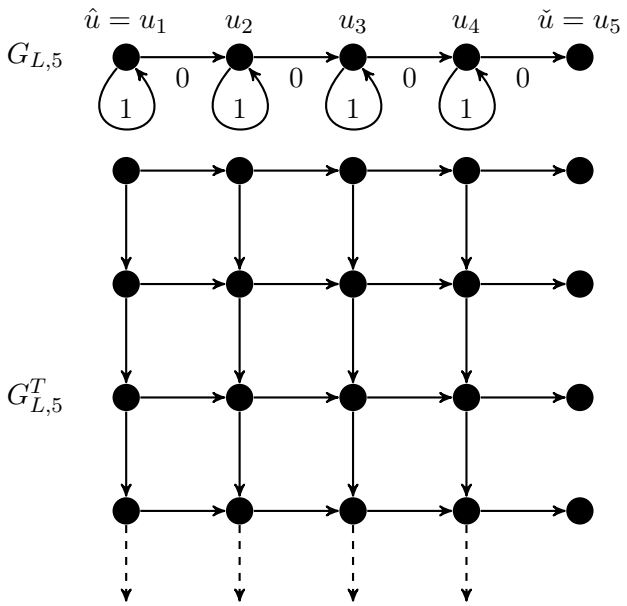


Figure 4.2: The grid graph $G_{L,n}$ and its time expansion $G^T_{L,n}$ for $n = 5$.

Example 4.9 Assume (C1) does not hold and set $h = 1$. Suppose G^{cur} is a valid subnetwork. Choose a path $P^* \in \mathcal{P} \setminus \mathcal{P}^{\mathsf{act}}$ and $a' \in A^T(\mathcal{P}) \setminus A^{\mathsf{act}}$. Let h^{aug} be

$$h^{\mathsf{aug}}(a) = \begin{cases} -(h(P^*)+1), & \text{if } a = a', \\ 0, & a \in A^{\mathsf{act}}, \\ 1, & \text{otherwise.} \end{cases}$$

Note that h and h^{aug} fulfil (C2) and (C3). For each path $P \in \mathcal{P}^{\mathsf{act}}$ there holds $h^{\mathsf{cur}}(P) = h^{\mathsf{aug}}(P) = 0$ and for each path $P \in \mathcal{P}^{\mathsf{cur}} \setminus \mathcal{P}^{\mathsf{act}}$ there holds $h^{\mathsf{cur}}(P) > 0$, because such a path contains at least one arc $\bar{a} \in A^T(P) \setminus A^{\mathsf{act}}$ with $h^{\mathsf{cur}}(\bar{a}) = h(\bar{a}) = 1$. However, $h^{\mathsf{aug}}(P^*) < 0$. So the shortest path in G^{cur} w. r. t. h^{cur} is contained in A^{act}, but it is not a shortest path in G^T w. r. t. h^{aug}. Hence, G^{cur} is not a valid subnetwork.

Remark 4.10 Note that if (C3) holds h^{aug} differs only on a finite number of arcs from h. Thus it is always possible to satisfy (C1) by simply including all arcs $a \in A^T$ with $h^{\mathrm{aug}}(a) < h(a)$ in G^{act} (of course, one can do this in advance and enlarge G^{act} without requiring a single solution of $(DyGG^{\mathrm{cur}}(G^{\mathrm{cur}},h^{\mathrm{aug}}))$).

Remark 4.11 Condition (C1) is naturally met in our framework if all coupling constraints are inequalities with non-negative coefficients, *i. e.*

$$Cx \leq b, C \geq 0.$$

In this case the associated Lagrange multipliers y are non-negative, so the active objective function fulfils

$$h^{\text{aug}}(x) = h(x) + \langle C^T y, x \rangle \geq h(x).$$

In the presence of negative coefficients or equality constraints this relation may not hold in general. For our practical application, the TTP, we investigate this condition in detail in Section 4.5.

Whereas the first condition is necessary to guarantee the existence of valid subnetworks, the other two conditions ensure that Algorithm 4.1 requires only a finite number of iterations. The following example shows that the algorithm may not stop after a finite number of iterations if conditions (C2) or (C3) do not hold.

Example 4.12 Consider again the grid graph $G^T_{L,2}$ from Example 4.8 and the basic objective function

$$h((u,t_u)(v,t_v)) = \begin{cases} 1/t_v, & \text{if } uv = u_1u_2, \\ 0, & \text{otherwise,} \end{cases}$$

and $h^{\text{aug}} = h$. Clearly conditions (C1) and (C3) are fulfilled but (C2) is not. Furthermore there exists no shortest path w. r. t. $h^{\text{aug}} = h$, hence Algorithm 4.1 would run forever.

Now consider the case $n = 2$, the basic objective function $h = 0$ and the active objective function as above, *i. e.*

$$h^{\text{aug}}((u,t_u)(v,t_v)) = \begin{cases} 1/t_v, & \text{if } uv = u_1u_2, \\ 0, & \text{otherwise.} \end{cases}$$

In this case (C1) and (C2) are fulfilled but (C3) is not. As in the previous example Algorithm 4.1 will not stop after a finite number of iterations.

The reason why this type of counter-examples works is that subproblem (DyGG(h^{aug})) to be solved w. r. t. h^{aug} does not have an optimal solution. The following lemma verifies that the validity of the conditions guarantees the existence of a finite optimal solution.

Lemma 4.13 *Assume (C2) and (C3) hold, then (DyGG(h^{aug})) has a finite optimal solution.*

Proof. Choose $\hat{t} := \max\{t_v \colon a = (u,t_u)(v,t_v) \in A^T, h^{\text{aug}}(a) \neq h(a)\} + 1$, which is possible by (C3), and set $\bar{d} := \max \bigcup_{a \in A} d(a)$. By the definition of the time expansion G^T each path from $\mathcal{P}$ containing a node (v,t_v) with $t_v > \hat{t} + \bar{d}$ contains some node (u,t_u) with $\hat{t} \leq t_u \leq \hat{t} + \bar{d}$. Furthermore we define $\mathcal{R}_u$, $u \in V$, to be the (finite) set of all paths of G of the form $u \dots \breve{u}$.

Now choose for each $(u,t_u) \in V \times \{\hat{t}, \dots, \hat{t} + \bar{d}\}$ and each $R \in \mathcal{R}_u$ a path $P_{u,t_u,R} \subset G^T$ with $P_{u,t_u,R} = (u,t_u)P_{u,t_u,R}$ and $(P_{u,t_u,R})_{|G} = R$, *i. e.* the path starts in (u,t_u) and continues along the nodes of R. Set

$$\tilde{t} := \max\left(\left\{\hat{t}\right\} \cup \left\{\bar{t}(P_{u,t_u,R}) \colon (u,t_u) \in V \times \{\hat{t}, \dots, \hat{t} + \bar{d}\}, R \in \mathcal{R}_u\right\}\right).$$

The finite subgraph $\tilde{G}^T = G^T[\{(u,t_u) \in V^T \colon t_u \leq \tilde{t}\}]$ contains a shortest path in $\mathcal{P}$ w. r. t. h^{aug}: Let $Q = Q(\breve{u}, t_{\breve{u}}) \in \mathcal{P}$ be a path with $t_{\breve{u}} > \tilde{t}$ and set $R := Q_{|G}$. Then Q contains a node $(u,t_u) \in V \times \{\hat{t}, \dots, \hat{t} + \bar{d}\}$. By (C2) and the choice of $\tilde{t} \geq \hat{t}$ it holds

$$h^{\text{aug}}((u,t_u)Q) = h((u,t_u)Q) \geq h(P_{u,t_u,R}) = h^{\text{aug}}(P_{u,t_u,R}),$$

hence $h^{\text{aug}}(Q(u,t_u)P_{u,t_u,R}) \leq h^{\text{aug}}(Q)$ and $Q(u,t_u)P_{u,t_u,R} \subset \tilde{G}^T$. □

However, even the existence of finite optimal solutions for both problems, the shortest path problem w. r. t. h resp. h^{aug}, is not sufficient for Algorithm 4.1 to be finite.

Example 4.14 Consider the grid graph $G^T_{L,2}$ from Example 4.12 and the cost functions

$$h((u,t_u)(v,t_v)) = \begin{cases} 0, & \text{if } uv = u_1u_2 \text{ and } t_u = t_v = 1, \\ 1, & \text{otherwise,} \end{cases}$$

and

$$h^{\text{aug}}((u,t_u)(v,t_v)) = \begin{cases} 2, & \text{if } uv = u_1u_2 \text{ and } t_u = t_v = 1, \\ 3, & \text{otherwise.} \end{cases}$$

Clearly these objective functions satisfy all conditions except (C3). Assume the active subgraph is given by $G^{\text{act}} = G^T_{L,2}[\{(u,t) \in V^T_{L,2} \colon t \leq \bar{t}\}]$ for some fixed time step $\bar{t} \in T$ and the corresponding current subgraph has been constructed as $G^{\text{cur}} = G^T_{L,2}[\{(u,t) \in V^T_{L,2} \colon t \leq \bar{t} + 1\}]$. Then the shortest path of the restricted subproblem is $P = (u_1,1) \dots (u_1,\bar{t})(u_2,\bar{t})$ with $h^{\text{cur}}(P) = 1$ and $P \nsubseteq G^{\text{act}}$, thus this path is not accepted as shortest path. Consequently G^{act} is increased by P, *i. e.* by one time step, and the next iteration is executed, which in turn increases G^{cur} by one time step and so on.

This example shows that the finiteness of the algorithm may depend on the actual choice of the current subnetwork. For example, if the current subnetwork in the previous example had been chosen as $G^{\text{cur}} = G^{\text{act}}$ then the subproblem would immediately return the correct path $P = (u_1, 1)(u_2, 1) \subset G^{\text{act}}$ as solution (G^{cur} is valid according to Observation 4.6). The next lemma shows that under our assumptions the finiteness of Algorithm 4.1 is independent from the concrete choice of the current subgraph (as long as it is a valid subnetwork).

Lemma 4.15 *Assume (4.2),(C2) and (C3) hold, then Algorithm 4.1 stops in finite time.*

Proof. We prove this by contradiction, so assume that the algorithm requires an infinite number of iterations and denote the returned shortest path of iteration $i \in \mathbb{N}$ by P^i. Analogously to Lemma 4.13 define

$$\hat{t} := \max\{t_v \colon a = (u, t_u)(v, t_v) \in A^T, h^{\text{aug}}(a) \neq h(a)\} + 1$$

(well defined by (C3)) and set $\bar{d} := \max \bigcup_{a \in A} d(a)$. By the definition of the time expansion G^T each path from $\mathcal{P}$ containing a node (v, t_v) with $t_v > \hat{t} + \bar{d}$ contains some node (u, t_u) with $\hat{t} \leq t_u \leq \hat{t} + \bar{d}$.

Let $\mathcal{R}_u$, $u \in V$, be the (finite) set of all paths of G of the form $u \dots \breve{u}$. Define for $(u, t_u) \in V^T$ and $R \in \mathcal{R}_u$ the index

$$i((u, t_u), R) := \min\{i \in \mathbb{N} \colon (u, t_u) \in V^T(P^i), ((u, t_u)P^i)_{|G} = R\}$$

the index of the *first* path among the P^i that visits (u, t_u) and continues along the nodes of R (if no such path exists then $i((u, t_u), R) = \infty$). Using (C2) we set

$$\tilde{t} := \max\{\bar{t}(P^i) \colon u \in V, \hat{t} \leq t_u \leq \hat{t} + \bar{d}, R \in \mathcal{R}_u, i = i((u, t_u), R) < \infty\}.$$

We claim that no path P^i contains a node (v, t_v) with $t_v > \tilde{t}$. Assume for contradiction that $P^{\hat{\imath}}$ contains a node (v, t_v) with $t_v > \tilde{t}$. Let $(u, t_u) \in V^T(P^{\hat{\imath}})$ be a node with $\hat{t} \leq t_u \leq \hat{t} + \bar{d}$, which exists by the consideration above. Because $d(a) \subset \mathbb{N}_0$ for all $a \in A$ we know by $t_u < t_v$ that (u, t_u) precedes (v, t_v) in $P^{\hat{\imath}}$, *i. e.* $(v, t_v) \in V^T((u, t_u)P^{\hat{\imath}})$. Set $\hat{R} := ((u, t_u)P^{\hat{\imath}})_{|G} \in \mathcal{R}_u$ and set $\hat{\jmath} = i((u, t_u), \hat{R})$. The path $P^{\hat{\imath}}$ contains the node (v, t_v) with $t_v > \tilde{t}$, so by the definition of $\tilde{t}$ we have $P^{\hat{\imath}} \neq P^{\hat{\jmath}}$, thus $\hat{\jmath} < \hat{\imath}$. This means $P^{\hat{\jmath}}$ has been the shortest path of an earlier iteration and therefore is contained in $G^{\text{cur}} \supseteq G^{\text{act}}$ when $P^{\hat{\imath}}$ is being computed. Finally $t_u \geq \hat{t}$, hence by the choice of $\hat{t}$ it holds

$$h^{\text{cur}}((u, t_u)P^{\hat{\jmath}}) = h^{\text{aug}}((u, t_u)P^{\hat{\jmath}}) = h((u, t_u)P^{\hat{\jmath}})$$

and

$$h^{\text{cur}}((u, t_u)P^{\hat{\imath}}) = h^{\text{aug}}((u, t_u)P^{\hat{\imath}}) = h((u, t_u)P^{\hat{\imath}}),$$

and $t_v > \tilde{t}$ implies by (C2)

$$h((u,t_u)P^{\hat{j}}) \leq h((u,t_u)P^{\hat{i}}).$$

Putting all together the path $Q := P^{\hat{i}}(u,t_u)P^{\hat{j}}$ fulfils $h^{\text{cur}}(Q) \leq h^{\text{cur}}(P^{\hat{i}})$ and $Q \subset G^{\text{cur}}$. Hence, Q is also a shortest path in this iteration and by (4.2) (Q arrives at $\check{u}$ earlier than $P^{\hat{i}}$) had been returned instead of $P^{\hat{i}}$, a contradiction.

We have shown that no path P^i, $i \in \mathbb{N}$, can contain a node at some time step greater than $\tilde{t}$. But there is only a finite number of such paths, hence the algorithm has to return one path, say P', a second time after a finite number of iterations. The second time this path is returned it is contained in G^{act} and the algorithm stops, hence the algorithm must stop after a finite number of iterations. □

Conditions (C2) and (C3) are rather artificial, though relatively weak. Indeed, different conditions are possible as long as they guarantee that a shortest path w. r. t. h and h^{aug} exists within a bounded time horizon. The following condition is another example, which is also quite useful in practical applications.

(C2') *(growth property):*
for each $\gamma \in \mathbb{R}$ there is a time threshold $\bar{t}_\gamma \in T$ so that for all $P = (u_1,t_1)\dots(u_k,t_k)$ in G^T with $u_k = \check{u}$ and $t_k \geq \bar{t}_\gamma$ there holds $h(P) \geq \gamma$.

This condition means that paths that have a sufficiently large final arrival time will eventually have arbitrarily large costs. An example is any objective function with $h \geq 0$ and $h((u,t_u)(\check{u},t_{\check{u}})) = \log t_{\check{u}}$. In this case the proof of finiteness is even simpler.

Lemma 4.16 *Assume (C1) and (C2') hold, then (DyGG(h^{aug})) has a finite optimal solution and Algorithm 4.1 stops after a finite number of iterations.*

Proof. We consider the first iteration of Algorithm 4.1. Let $P \in \mathcal{P}^{\text{act}}$ be an arbitrary path. By assumption the active subgraph G^{act} is finite, thus by (C1) and (C2') there is a time step $\hat{t} \in T$ so that $\hat{t} \geq \max\{t\colon (u,t) \in V^T(P)\}$, and each path P' containing a node $(v,t_v) \in V^T(P')$ with $t_v \geq \hat{t}$ has higher costs

$$h(P) \leq h(P') \qquad \text{and} \qquad h^{\text{aug}}(P) \leq h^{\text{aug}}(P'),$$

and consequently

$$h^{\text{cur}}(P) \leq h^{\text{cur}}(P').$$

Note that these relations hold for *all* iterations, because h^{aug} does not change (only G^{act} grows). Let $G' = G^T[\{(u,t_u)\colon t_u \leq \hat{t}\}]$. This subnetwork is finite and contains the shortest path of G^T w. r. t. h^{aug}, so (DyGG(h^{aug})) has a finite optimal solution. Assume Algorithm 4.1 performs an infinite number of iterations. The shortest paths computed in each iteration are also contained in G' (because G^{act} only grows, thus $P \subset G^{\text{act}}$). Because there is only a finite number of paths contained in G', some path, say $P^* \in \mathcal{P}$, must eventually be returned a second time. When this happens P^* is contained in G^{act} and the algorithm stops. □

Remark 4.17 An example of an objective function that does not satisfy (C2') but may satisfy (C2) and (C3) and may be useful in practice is $h = 0$. This objective function may occur if the network does not contribute any costs but carries some structural information about the problem. An example of such networks are the configuration networks described in Section 2.6, which only model when a certain track is opened for a train. But note that configuration networks do not fit in our framework because their respective base graphs are not acyclic (but see Section 4.5).

4.2.2 Basic Approach

If the cost function h^{aug} changes arbitrarily on the whole graph in each single invocation of (DyGG(h^{aug})), then one cannot expect an efficient algorithm that only works on some subgraph of G^T. This is also true if the augmented cost function h^{aug} has small values (smaller than h) in arcs outside the active subgraph G^{act}. In this case the shortest path in G^T w. r. t. h^{aug} is likely not contained in G^{act} and the subgraph has to be enlarged anyway (this is the reason for (C1)). Fortunately the cost functions h^{aug} do not change completely arbitrarily in the algorithmic framework within which we want to apply dynamic graph generation, namely a bundle or subgradient method applied to a Lagrangian relaxation of a planning problem.

From now on we assume that the conditions (C1)–(C3) hold for all cost functions h and h^{aug}. By property (C1) we know that h always acts as a lower bound on h^{aug} on arcs outside G^{act}. In particular, by definition (4.1) of h^{cur} this property transfers nicely to the current cost function h^{cur} that is used in ($DyGG^{\text{cur}}(G^{\text{cur}},h^{\text{aug}})$).

Observation 4.18 *For h^{aug} satisfying (C1) there holds*

(C1') $\forall\, a \in A^T \colon h^{cur}(a) \leq h^{aug}(a)$.

The point of using h^{cur} instead of h^{aug} in subproblem ($DyGG^{\text{cur}}(G^{\text{cur}},h^{\text{aug}})$) is that h^{cur} carries the structural properties of h on arcs outside G^{act}. While the structure of h^{aug} may change unpredictably during the solution of the Lagrangian dual problem, the structure of h is known in advance. Therefore the idea is to exploit special structural properties of h to construct valid subnetworks G^{cur}.

The next question is how large G^{cur} must be chosen in order to exploit the structural properties of h. The goal is to choose G^{cur} just large enough so that it contains a shortest path of G^T w. r. t. h^{cur}.

Observation 4.19 *If there exists a shortest path $P \in \mathcal{P}$ w. r. t. h^{cur} in G^T with $P \in \mathcal{P}^{cur}$, then G^{cur} is a valid subnetwork.*

Proof. Let $P^* \in \mathcal{P}^{\text{cur}}$ be a shortest path w. r. t. h^{cur} in G^{cur} and assume $P^* \subset G^{\text{act}}$. Let $P' \in \text{Argmin}\{h^{\text{aug}}(P) \colon P \in \mathcal{P}\}$. By assumption we have $h^{\text{cur}}(P^*) \leq h^{\text{cur}}(P')$ and by definition of h^{cur} and $P^* \subset G^{\text{act}}$ we have $h^{\text{cur}}(P^*) = h^{\text{aug}}(P^*)$. Furthermore by (C1') we know $h^{\text{cur}}(P') \leq h^{\text{aug}}(P')$, hence

$$h^{\text{aug}}(P^*) = h^{\text{cur}}(P^*) \leq h^{\text{cur}}(P') \leq h^{\text{aug}}(P') \leq h^{\text{aug}}(P^*)$$

and $P^* \in \text{Argmin}\{h^{\text{aug}}(P) \colon P \in \mathcal{P}\}$. Thus G^{cur} is a valid subnetwork. □

In order to construct G^{cur} large enough we connect the structural properties of h and those of the time expanded network G^T. The nodes that lie on the boundary of the active subgraph G^{act} will be of special interest.

Definition 4.20 The set of *boundary nodes* of G^{act} is defined by

$$\partial A^{\text{act}} := \{u \in V^{\text{act}} \colon \exists\, (u,v) \in A^T \setminus A^{\text{act}} \vee \exists\, (v,u) \in A^T \setminus A^{\text{act}}\}.$$

∂A^{act}-path A ∂A^{act}-path is a path $P = (u,t_u)\dots(v,t_v) \subset A^T \setminus A^{\text{act}}$ with $(u,t_u) \in V^{\text{act}}$ and $(v,t_v) \in V^{\text{act}} \cup (\{\check{u}\} \times T)$.

If the following holds, then this establishes a connection between h and G^{cur}:

(S) for each ∂A^{act}-path $P = (u,t_u)P(v,t_v)$ there is a ∂A^{act}-path $P' = (u,t_u)P'(v,t'_v) \subset A^{\text{cur}}$ with $v = \check{u} \vee t'_v = t_v$, and P' fulfils $h(P') \leq h(P)$.

Condition (S) is in fact a structural condition on the objective function h *and* therefore on the objective function h^{cur}: it implies that for any path that leaves G^{cur} there is a path contained in G^{cur} with the same start node and end node (or arriving at $\check{u}$) that has no more expensive objective value w. r. t. h. We will prove now that a network satisfying (S) is indeed a valid subnetwork.

Lemma 4.21 *If G^{cur} satisfies (S) then G^{cur} is a valid subnetwork.*

Proof. We will prove that G^{cur} satisfies the condition of Observation 4.19. Assume, for contradiction, that there exists no path in $\mathcal{P}^{\text{cur}}$ that is a shortest path in G^T w. r. t. h^{cur}.

Let $P \in \mathcal{P}$ be a shortest path in G^T w. r. t. h^{cur} so that the number of arcs $|A^T(P) \setminus A^{\text{cur}}|$ not contained in A^{cur} is as small as possible. Because $P \nsubseteq G^{\text{cur}}$ this number is greater than zero. So there must be nodes $(u, t_u) \in V^T(P) \cap \partial A^{\text{act}}$ and $(v, t_v) \in V^T(P) \cap (\partial A^{\text{act}} \cup (\{\check{u}\} \times T))$ so that $A^T((u, t_u)P(v, t_v)) \cap A^{\text{act}} = \emptyset$ and $A^T((u, t_u)P(v, t_v)) \nsubseteq A^{\text{cur}}$, *i. e.* $(u, t_u)P(v, t_v)$ is a ∂A^{act}-path leaving G^{cur}. Condition (S) implies that there is a path $Q = (u, t_u)Q(v, t'_v) \subset G^{\text{cur}}$ with $h(Q) \leq h((u, t_u)P(v, t_v))$ and $t'_v = t_v$ if $v \neq \check{u}$. We set

$$P' := \begin{cases} P(u, t_u)Q, & \text{if } v = \check{u}, \\ P(u, t_u)Q(v, t_v)P, & \text{otherwise.} \end{cases}$$

The path P' is contained in $\mathcal{P}$, contains less arcs than P that are not in A^{cur} and fulfils $h(P') \leq h(P)$, a contradiction. The assertion follows from Observation 4.19. □

Property (S) gives a sufficient condition for a current subnetwork G^{cur} to be valid. In the next sections we will develop possible constructions of such a network. But first we give a simple example to illustrate how a subnetwork fulfilling (S) can be constructed.

Example 4.22 Consider the grid graph $G^T_{L,n}$ from Example 4.8 and define the cost function

$$h\colon A^T \to \mathbb{R}, h((u, t_u)(v, t_v)) = \begin{cases} t_v & \text{if } v = \check{u}, u \neq v, \\ 0 & \text{otherwise.} \end{cases}$$

This means the cost of some schedule depends only on the end time of the final step (which could, *e. g.*, be the completion time of the production of some object or the arrival time of a train at its final station). Let $G^{\text{act}} \subset G^T_{L,n}$ be a finite subgraph and set $t_{\max} := \max\{t_u\colon (u, t_u) \in V^{\text{act}}\}$. Then the graph

$$G^{\text{cur}} := \text{cl}\{(u, t_{\max} + 1)\colon u \in V\}$$

is a valid subnetwork (see Figure 4.3a).

Proof. Let P be an arbitrary ∂A^{act}-path. If $P \nsubseteq G^{\text{cur}}$ then it must contain a node $(u_i, t_{\max} + 1)$, $i \in \{1, \dots, n\}$ with i as small as possible. We cannot have $i = n$, because $u_n u_n \notin A_{L,n}$ implies $(u_{n-1}, t_{\max} + 1)(u_n, t_{\max} + 1) \in A_{L,n}(P)$ contradicting the choice of i. Because all traversal times are non-negative we know that each node (v, t_v) that succeeds $(u_i, t_{\max} + 1)$ in P fulfils $t_v \geq t_{\max} + 1$, thus P has the form

$$P = P(u_i, t_{\max} + 1)P(u_{n-1}, t_{u_{n-1}})(\check{u}, t_{\check{u}})$$

with $t_{\check{u}} \geq t_{\max} + 1$ and $h(P) = t_{\check{u}}$. By definition of G^{cur} the path

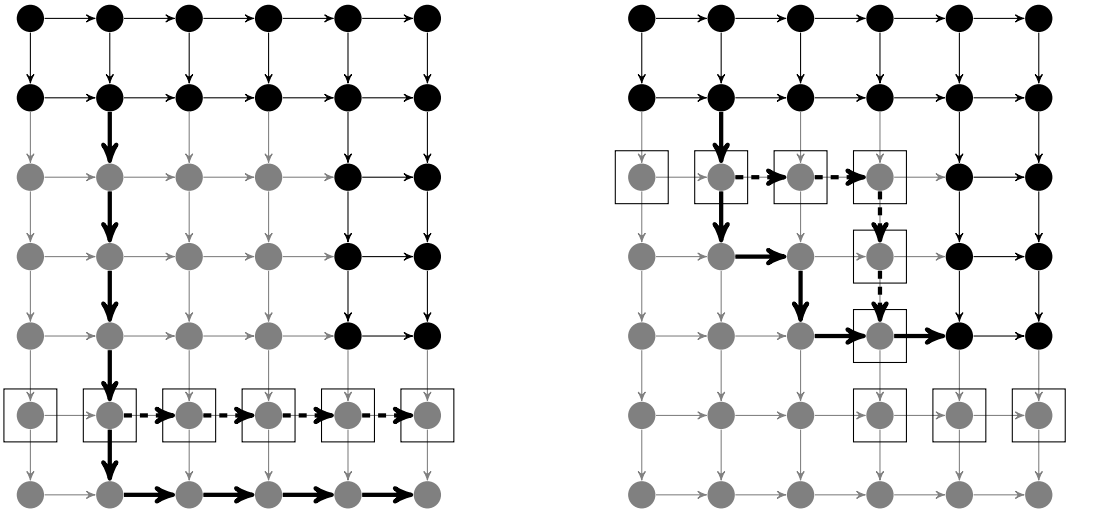
$$Q = (u_i, t_{\max} + 1)(u_{i+1}, t_{\max} + 1) \dots (u_n, t_{\max} + 1) \subset G^{\text{cur}},$$

is contained in G^{cur}, thus the path

$$P' := P(u_i, t_{\max} + 1)Q(\check{u}, t_{\max} + 1)$$

is a ∂A^{act}-path with arrival time $t_{\max} + 1$ at the final node and therefore

$$h(P') = t_{\max} + 1 \leq t_{\check{u}} = h(P).$$ □



(a) Simple corridor. The grey nodes between the rectangles and the black nodes belong to G^{cur}, too.

(b) Shifted corridor. G^{cur} consists precisely of the black nodes and the nodes within the boxes.

Figure 4.3: Example of a valid subnetwork for $G^T_{L,n}$. The black nodes are the active subgraph G^{act}. The current subnetwork G^{cur} consists of all nodes surrounded by boxes and all nodes above them (by the closure operation). The thick path is a ∂A^{act}-path, the dashed path its replacing path.

This example uses a very simple construction of the current subnetwork: Using a "horizontal" path as close as possible to the active subnetwork. This path contains a valid continuation (because it is a path). Although simple, Figure 4.3a shows a drawback of this approach: if the boundary of the active subgraph is not "flat", *i. e.* contains some wait nodes and arcs, then the constructed G^{cur} may contain a lot of unnecessary nodes. This suggests the following improvement.

Example 4.23 Consider the same graph and objective function as before in Example 4.22 Let $G^{\mathsf{act}} \subset G^T_{L,n}$ be a finite subgraph. Then the graph

$$G^{\mathsf{cur}} := \operatorname{cl}\left\{(u_i, \bar{t}_i) \in V^T_{L,n} \colon i \in \{1, \ldots, n\}\right\}$$

where

$$\bar{t}_i := \max\{t + 1 \colon (u_j, t) \in V^{\mathsf{act}}, j \leq i + 1\} \tag{4.3}$$

is a valid subnetwork (see Figure 4.3b).

Proof. First observe that by definition of the objective function it holds

$$P' \leq_T P \Rightarrow h(P') \leq_T h(P),$$

so it suffices to show that for each ∂A^{act}-path P there is a replacing path P' with $P' \leq_T P$.

Now we show that $G^{\text{cur}} \setminus G^{\text{act}}$ contains a path over all $u_i \in V_{L,n}$. Indeed, set

$$R = (u_1, \bar{t}_1)(u_2, \bar{t}_1) \dots (u_2, \bar{t}_2)(u_3, \bar{t}_2) \dots (u_3, \bar{t}_3) \dots (u_n, \bar{t}_{n-1}) \dots (u_n, \bar{t}_n).$$

Note that $t_n = \bar{t}_n$ because $u_n u_n \notin A_{L,n}$. Then $R \subset G^{\text{cur}} \setminus G^{\text{act}}$, because by (4.3) it holds

(i) $\bar{t}_{i-1} \leq \bar{t}_i$ for all $i \in \{2, \dots, n\}$,

(ii) $(u_i, \bar{t}_i) \in V^{\text{cur}} \setminus V^{\text{act}}$ for all $i \in \{1, \dots, n\}$,

(iii) $(u_i, \bar{t}_{i-1}) \in V^{\text{cur}} \setminus V^{\text{act}}$ for all $i \in \{2, \dots, n\}$, and

(iv) $(u_{i-1}, \bar{t}_{i-1})(u_i, \bar{t}_{i-1}) \in A^{\text{cur}} \setminus A^{\text{act}}$ for all $i \in \{2, \dots, n\}$.

Let

$$P = (u_k, t_k) \dots (u_k, t_{k+1})(u_{k+1}, t_{k+1}) \dots (u_{k+1}, t_{k+2}) \dots (u_{\bar{k}}, t_{\bar{k}})$$

be an arbitrary ∂A^{act}-path with $P \nsubseteq G^{\text{cur}}$. Then by (i) there must be a smallest i so that $(u_i, \bar{t}_i) \in V^T_{L,n}(P)$ and $(u_i, \bar{t}_i + 1) \in V^T_{L,n}(P)$. We consider two cases

a) Assume $t_j > \bar{t}_j$ for all $j \in \{i+1, \dots, \bar{k}\}$, *i. e.*, P never returns to G^{cur} (and to G^{act}). Because P is a ∂A^{act}-path we have $\bar{k} = n$, thus the path

$$P' = P(u_i, \bar{t}_i)R(u_n, \bar{t}_n) \dots (u_n, t_k)$$

is a ∂A^{act}-path and fulfils $P' \leq_T P$.

b) Otherwise there is a smallest $\bar{\imath} \in \{i+1, \dots, k\}$ with $t_{\bar{\imath}} \leq \bar{t}_{\bar{\imath}}$. Then the path

$$P' = P(u_i, \bar{t}_i)R(u_{\bar{\imath}}, \bar{t}_{\bar{\imath}})P$$

is a ∂A^{act}-path with $P' \leq_T P$ and

$$|V^T_{L,n}(P') \setminus V^{\text{cur}}| < |V^T_{L,n}(P) \setminus V^{\text{cur}}|$$

(because $(u_i, \bar{t}_i + 1) \in V^T_{L,n}(P) \setminus V^{\text{cur}}$ but $R \subseteq G^{\text{cur}}$). Iterating the argument we end up with a ∂A^{act}-path $P'' \leq_T P$ and $|V^T_{L,n}(P'') \setminus V^{\text{cur}}| = 0$, so P'' satisfies (S). □

Note that these examples would work, too, if the time expansion G^T is a *planar* graph, because then there exists a path R as in Example 4.23 that *separates* G^{act} from the rest of the graph. Here separating means that any path the crosses R must actually share a

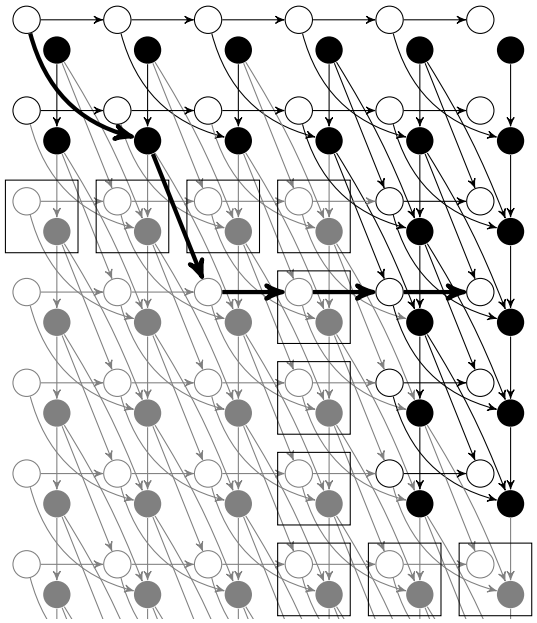


Figure 4.4: Non planar time expansion. If the transition times are so that the time expansion G^T is non-planar, then choosing the nodes within the boxes as generators for G^{cur} similar to Example 4.23 is *not* sufficient: the thick path has no replacing path, because it "jumps" over the corridor formed by the boxes.

node with R. If the time expansion is not planar, then such a separating path R does not exist in general, see Figure 4.4.

Nevertheless these simple examples motivate the next steps. Instead of using a single separating path, we construct a bunch of paths that separate G^{act} from the rest, that we will call a "corridor". This corridor must have the following properties, which automatically hold in the grid-graph examples above. First the corridor must live outside of G^{act}, so we have control over the objective value (outside G^{act} the cost function h^{cur} equals h). Second we have to ensure that we can "redirect" paths that leave G^{cur} into the corridor. Third, even if G^{cur} is large enough so that for any ∂A^{act}-path P there is some ∂A^{act}-path P' with the same start and end node that is contained in G^{cur} this does not imply immediately that $h^{\text{cur}}(P') \leq h^{\text{cur}}(P)$. We need some additional structural properties of the original objective function h that allow us to conclude that relation from some relation between the original path P and its replacing path P'.

In the next section we start constructing a current subnetwork G^{cur}, which resembles Example 4.22. This subnetwork will be based on a "fastest corridor", which will contain the fastest possible continuations for any replacing path. This corridor is then moved as close as possible to the active subgraph by choosing an appropriate *common* time shift for all its nodes. As in Example 4.22, this corridor may be quite far away from G^{act} if the boundary of the active subgraph does not follow the fastest continuations.

Thus in Section 4.4 we will modify the corridor by shifting some parts of it closer to G^{act} exploiting wait arcs similar to Example 4.23.

4.3 A Valid Subnetwork for Complex Cost Structures

The first step in order to determine an appropriate current subgraph G^{cur} is to specify suitable structural properties of the original cost function h. Example 4.22 motivates the following requirements:

> A path arriving at the final node at some later time step must not be cheaper than one arriving at an earlier time step.

As mentioned above, this may indeed be satisfied in some practical applications if the primary objective is the completion time of some production process or the arrival time at the final station. Also note that this property is similar to (C2), which we assume to be satisfied anyway, but stronger. Another possible property could be:

> If P, Q are two paths in G^T that follow the same path in G but P runs "earlier" than Q at each node, then $h(P) \leq h(Q)$.

Again, a cost function like this seems "natural" for many applications because a schedule that is earlier than another one at each single step is often preferable. This informal property has two shortcomings. First we do not know what "earlier" means and second the two paths P and Q must follow the same route in G. We will formalise both terms in the following.

The basic property of the cost function will be defined along paths in the clone graph $[G]$ of G, not along single paths in G. This allows to compare paths with the same route even if they use different nodes of G. If two paths $P, Q \subseteq G^T$ use the same sequence of clone nodes, *i. e.*, $[P] = [Q]$, they can be compared regarding the time-steps at which they visit the single clone nodes.

Definition 4.24 Let

$$P = (u_1, t_1) \dots (u_1, t'_1)(u_2, t_2) \dots (u_2, t'_2)(u_3, t_3) \dots (u_k, t'_k)$$

and

$$Q = (v_1, s_1) \dots (v_1, s'_1)(v_2, s_2) \dots (v_2, s'_2)(v_3, s_3) \dots (v_k, s'_k)$$

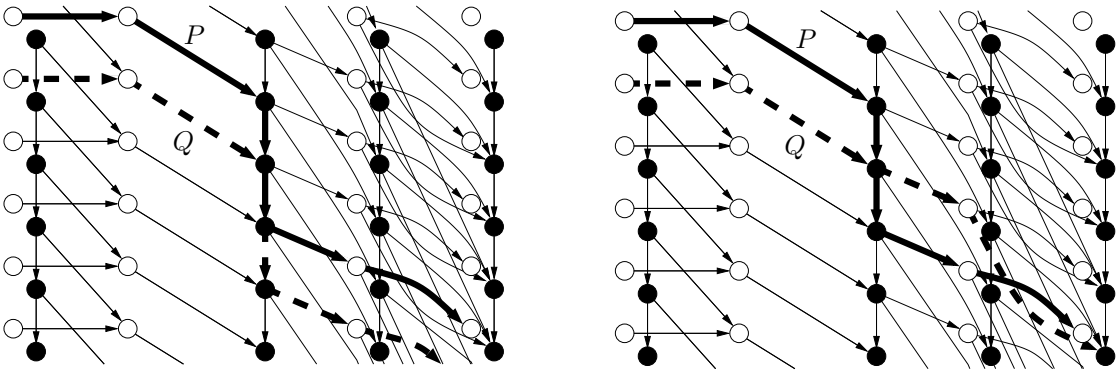
be two paths, $P, Q \subset G^T$. Then path P is *not later* than Q, write $P \leq_T Q$, if

(i) $[P] = [Q]$ and

(ii) $\forall\, i \in \{1, \dots, k\} \colon t_i \leq s_i \wedge t'_i \leq s'_i$.

P is *earlier* than Q, write $P <_T Q$, if $P \leq_T Q$ and $t_i < s_i \vee t'_i < s'_i$ for some $i \in \{1, \dots, k\}$.

So $P \leq_T Q$ if and only if P enters each clone node not later than Q and also leaves each clone node not later than Q. Note that it is perfectly possible to have $P \leq_T Q$ while there are nodes $(u, t_u) \in V^T(P)$ and $(u', t_{u'}) \in V^T(Q)$ with $[u] = [u']$ and $t_u > t_{u'}$ as long as condition (ii) holds, see Figure 4.5. Obviously, the relation $\leq_T$ introduces a partial ordering on the set of paths.



(a) $P \leq_T Q$ holds because P and Q follow the same (clone) route and P enters and leaves each clone node not later than Q.

(b) P and Q are not comparable w. r. t. $\leq_T$ because P enters some node earlier and some node later than Q.

Figure 4.5: The partial order $\leq_T$ of two paths P (solid arcs) and Q (dashed arcs).

Observation 4.25 *There is no infinite sequence of paths $(P^i)_{i \in \mathbb{N}}$ that decreases w. r. t. $<_T$, i. e., $P_1 >_T P_2 >_T \ldots$.*

Proof. $P^i > P^j$ implies that no entering and no leaving time of P^j is larger than the corresponding one of P^i and at least one entering or leaving time is smaller. Because all times are positive integers the sequence must be finite. □

The basic assumption on the original cost function h can now be formally stated as follows.

(C) For two paths $P, Q \subseteq A^T, P \leq_T Q \Rightarrow h(P) \leq h(Q)$.

From now on we will always assume that this property holds for original cost function h. Sometimes we assume that other properties hold instead, but we will always state this explicitly.

The following corollary explains the usefulness of property (C).

Corollary 4.26 *Given a network G^T, assume the original cost function h satisfies (C). Then a subgraph $G^{cur} \subset G^T$ with $G^{act} \subseteq G^{cur}$ is a valid subnetwork if the following condition holds*

(S') *for each ∂A^{act}-path $P = (u, t_u)P(v, t_v)$ there is a ∂A^{act}-path $P' = (u, t_u)P'(v, t'_v) \subset A^{cur}$ with $(v = \check{u} \vee t'_v = t_v)$ and $P' \leq_T P$.*

Proof. Follows directly from Lemma 4.21 and (C). □

The important point of this formulation is that it does not require the actual values of the cost function h anymore. Instead it relies solely on structural properties of the current subgraph G^{cur} w. r. t. G^{act}. The only property that we have to guarantee is that any path leaving G^{cur} can be replaced by an *earlier* path within G^{cur} (which will automatically be no more expensive). This motivates the following definition.

Definition 4.27 A ∂A^{act}-path $P = (u, t_u)P(v, t_v)$ is *G^{act}-reducible* if there is a ∂A^{act}-path $P' = (u, t_u)P'(v, t'_v)$ with $P' <_T P$ and $(v = \check{u} \vee t'_v = t_v)$. If no such path P' exists, P is called *G^{act}-irreducible*.

So G^{act}-irreducible paths are those ∂A^{act}-paths for which no better replacing path in $G^T \setminus G^{\text{act}}$ exists. Hence for G^{cur} to be a valid subnetwork it must contain at least all those paths. The following corollary states this formally.

Corollary 4.28 *Suppose G^{cur} contains G^{act} and all G^{act}-irreducible paths, then G^{cur} is a valid subnetwork whenever (C) holds.*

Proof. We want to apply Corollary 4.26, thus we have to show that (S') holds for G^{cur}. Let $P = P^1 = (u, t_u) \dots (v, t_v)$ be an arbitrary ∂A^{act}-path. If P^1 is irreducible then $P^1 \subset A^{\text{cur}} \setminus A^{\text{act}}$ and $P' = P^1$ suffices. So we may assume P^1 is reducible. But then, by definition, there is a ∂A^{act}-path P^2 with $P^2 <_T P^1$ with $P^2 = (u, t_u)P^2(v, t^2_v)$ and $(v = \check{u} \vee t^2_v = t_v)$. If P^2 is contained in A^{cur} we are done. Otherwise we may apply the same arguments iteratively and get a sequence $(P^i)_{i \in I \subseteq \mathbb{N}}$ of paths. By Observation 4.25 this sequence must be finite so its last path $P^{|I|}$ must be irreducible. Consequently $P' = P^{|I|}$ fulfils (S'). □

In view of Corollary 4.28 the purpose of the remainder of this chapter is to construct a valid subnetwork G^{cur} that contains all ∂A^{act}-irreducible paths. Because we allow the actual active subgraph G^{act} and the boundary ∂A^{act} to have arbitrary structure (as long as G^{act} is finite) it is usually not easy to identify all irreducible paths. Instead we look for properties that G^{cur} must have so that we can *construct* the replacing path of any path that leaves G^{cur}. We will identify difficult situations where such a replacement path cannot be found and increase G^{cur} appropriately.

The definition of irreducible paths suggests that they live in an area "near" the boundary of G^{act}: they are contained in $A^T \setminus A^{\text{act}}$ and there are no earlier such paths that can replace them. It will be convenient to think of that area as some kind of "corridor" G^{cor} along the boundary of G^{act} so that all irreducible paths are contained within that corridor. The corridor itself will not intersect with the active subgraph G^{act} and the current subnetwork will be constructed to contain this corridor, *i. e.* $G^{\text{cor}} \subset G^T \setminus G^{\text{act}}$ and $\operatorname{cl} G^{\text{cor}} \subseteq G^{\text{cur}}$. The goal is to construct G^{cor} large enough so that it contains all irreducible paths and at least one explicitly constructable replacing path for each reducible path. In particular, let $u, v \in V^T(P) \cap V^{\text{cur}}$ be two nodes of a reducible path P with $P = PuPvP$ and $uPv \nsubseteq G^{\text{cur}}$ then the part uPv of P is replaced by another part $Q = uQv \subset G^{\text{cur}}$ with $Q \subseteq G^{\text{cur}}$, $Q \leq_T uPv$, so that $P' = PuQvP$ is a replacing path of P. The corridor G^{cor} is used to identify appropriate nodes $u, v \in V^T(P)$ along with the path $Q = uQv \in G^{\text{cur}} \supset \operatorname{cl} G^{\text{cor}}$ so that this exchange can be done.

Having this strategy in mind, the corridor should be large enough to possess the following three properties.

1. It must have an *interception property*, *i. e.*, each path that crosses the corridor can be redirected into the corridor. Figure 4.6 shows a sketch of the active subgraph and the current subgraph with the corridor.

2. The corridor needs to satisfy a *continuation property*, *i. e.*, for every possible entry point into the corridor it has to contain a valid path that leads on to $\check{u}$ (because the longest reducible paths may go to $\check{u}$).

3. Finally, it must have a *reentrant property*, *i. e.*, for any path P that reenters G^{cor}, its redirection and continuation along the corridor must offer an alternative path $P' <_T P$ that meets P before or when P enters ∂A^{act}.

These three properties correspond to the three steps that are necessary to construct a replacing path: find a start node u to start the replacing path, a continuation $Q = uQv$ within the corridor and an end node v where the replacing path meets to original path again.

In principle it is not difficult to construct such a corridor. Indeed, as depicted in Figure 4.8a it suffices to determine in a preprocessing step a graph structure giving the fastest variant of each route in V towards $\check{u}$ with respect to *minimal arc times*

$$\underline{d}(u, v) := \min d((u, v)) \quad \text{for } (u, v) \in A.$$

For this, define a *t-shifted fastest route graph* $F^t = (V_F^t, A_F^t)$ for $t \in \mathbb{N}_0$ recursively by $(\check{u}, t) \in V_F^t$ and

$$\begin{aligned}&\left[uv \in A \wedge (u \neq v) \wedge (v, \tau) \in V_F^t\right] \\ &\qquad \Rightarrow \left[(u, \tau - \underline{d}(u, v)) \in V_F^t \wedge (u, \tau - \underline{d}(u, v))(v, \tau) \in A_F^t\right]. \quad (4.4)\end{aligned}$$

The graph F^t is well defined because G is acyclic (except loops) and posses the well defined partial order $\leq_G$ on the nodes V.

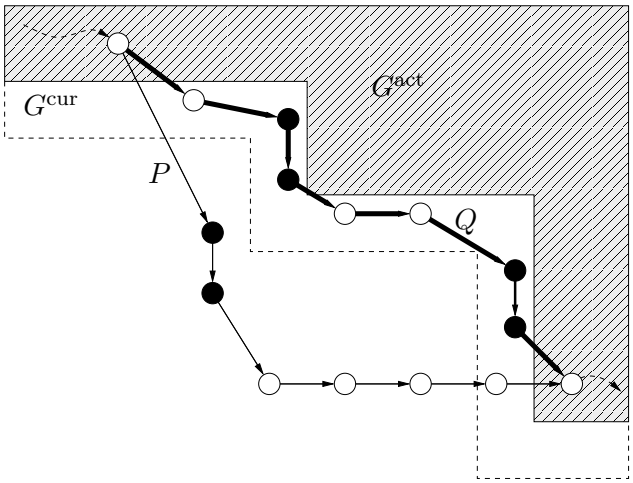


Figure 4.6: Active subgraph G^{act} (the hatched area) and current subgraph G^{cur}. The thin path P is redirected to the "corridor" in G^{cur}. In the corridor the thick path Q is the earlier replacement of the original path.

Remark 4.29 Note that F^t is only a subgraph of G^T if t is large enough, *i. e.* only if

$$\min\{t_u \colon (u, t_u) \in V_F^t\} \geq 1$$

by the definition of G^T.

The following important theorem states that the F^t are indeed sufficient to construct a valid subnetwork G^{cur}.

Theorem 4.30 *Let*

$$\underline{t} := \min\{t \in \mathbb{N}_0 \colon F^t \cap G^{act} = \emptyset\} \qquad \textit{and} \qquad \bar{d} := \max \bigcup_{e \in A} d(a).$$

With $G^{cor} := F^{\underline{t}+\bar{d}-1}$ the current subgraph $G^{cur} := \operatorname{cl} G^{cor}$ is a valid subnetwork whenever (C) holds.

Proof. First note that by definition (4.4)

$$(u, t_u) \in V_F^t \iff (u, t_u + 1) \in V_F^{t+1} \tag{4.5}$$

and in particular by definition of $\underline{t}$

$$F^{\underline{t}}, \ldots, F^{\underline{t}+\bar{d}-1} \subseteq \operatorname{cl} F^{\underline{t}+\bar{d}-1} = G^{\text{cur}},$$

i. e. all corridors F^t for $t \in \{\underline{t}, \dots, \underline{t} + \bar{d} - 1\}$ are contained in G^{cur}. Let

$$P = (u_1, \tau_1) \dots (u_1, \tau_1')(u_2, \tau_2) \dots (u_k, \tau_k') \subset A^T \setminus A^{\text{act}}$$

be a ∂A^{act}-path. If $P \subset G^{\text{cur}}$ there is nothing to show, so we assume $P \not\subset G^{\text{cur}}$. By the definition of F^t for each $t \in \mathbb{Z}$ there is a path

$$P^t = (u_1, \tau_1^t)(u_2, \tau_2^t) \dots (u_k, \tau_k^t) \subset F^t$$

where

$$\tau_{i+1}^t - \tau_i^t = \underline{d}(u_i u_{i+1}) \text{ for all } i \in \{1, \dots, k-1\}.$$

Because $(u_1, \tau_1) \in \partial A^{\text{act}}$ and $P \not\subset G^{\text{cur}}$ there must be a first node $(u_i, \tau) \in V^T(P)$ with $\tau \geq \tau_i^t$ and this node is not (u_1, τ_1), so it has a preceding node in P.

We show now that $(u_i, \tau) \in V^T(P^t)$ for some $t \in \{\underline{t}, \dots, \underline{t} + \bar{d} - 1\}$. On the one hand, if $\tau > \tau_i$ then $(u_i, \tau - 1) \in V^T(P)$ and by the choice of (u_i, τ) it is $\tau - 1 < \tau_i^{\underline{t}}$, thus $\tau = \tau_i^{\underline{t}}$ and $(u_i, \tau) \in V^T(P^{\underline{t}})$. On the other hand if $\tau = \tau_i$ then its preceding node in P is (u_{i-1}, τ_{i-1}') and again by the choice of (u_i, τ) it is $\tau_{i-1}' < \tau_{i-1}^{\underline{t}}$. Furthermore $\tau - \tau_{i-1}' \geq \underline{d}(u_{i-1}u_i)$ so we have

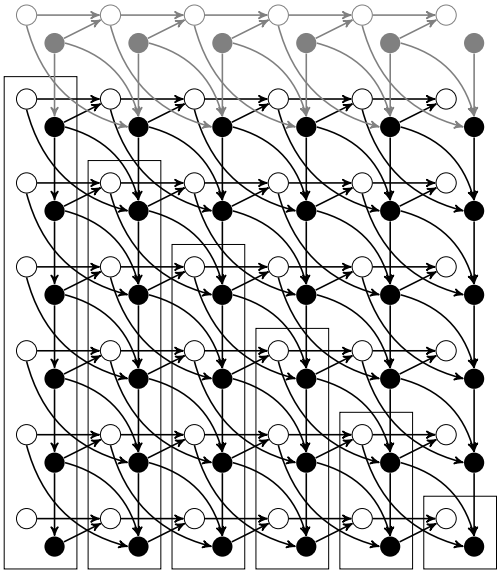
$$\tau_i^{\underline{t}} \leq \tau \leq \tau_{i-1}' + \bar{d} < \tau_{i-1}^{\underline{t}} + \bar{d} \leq \tau_i^{\underline{t}} + \bar{d}.$$

Consequently $(u_i, \tau) \in V^T(P^t)$ for some $t \in \{\underline{t}, \dots, \underline{t} + \bar{d} - 1\}$.

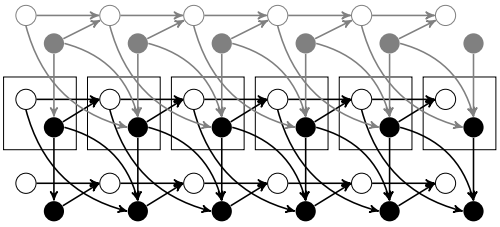
Now let $t \in \{\underline{t}, \dots, \underline{t} + \bar{d} - 1\}$ be so that $(u_i, \tau) \in V^T(P^t)$ and consider the path $P' := P(u_i, \tau)P^t$. Because $(u_i, \tau)P^t$ uses the same sequence of nodes $u_i, \dots, u_k$ but always continues using the fastest possible transition time $\underline{d}(u_j u_{j+1})$, $j \in \{i, \dots, k-1\}$, it is $P' \leq_T P$. Furthermore the choice of (u_i, τ) implies $P(u_i, \tau) \subset G^{\text{cur}}$ and by definition it holds $P^t \subset F^t \subset G^{\text{cur}}$, so $P' \subset G^{\text{cur}}$ as well. In particular, because $P \not\subset G^{\text{cur}}$, this proves $P' <_T P$. Thus P' is a valid replacing path of P and P is reducible. The result follows by Corollary 4.28. □

Looking at the proof we see that the definition of F^t includes more than a mere time expansion of the shortest path tree in $G^T = (V^T, A^T)$ in order to ensure the interception property and the continuation property along *any* route. The reentrant property is not needed here, because no path can outrun a fastest path along the same route (the path P leaving G^{cur} in the proof does indeed never return). While the structure of the correct F^t might be quite involved, there is actually no need to store or construct the complete structure of F^t. Indeed, due to the closure operation it suffices, *e. g.*, to construct and store for each node $u \in V$ the two values $\min\{\tau \colon (u, \tau) \in F^0\}$ and $\max\{\tau \colon (u, \tau) \in F^0\}$ relative to $(\check{u}, 0)$ within a preprocessing step, then G^{cur} is quickly formed for any given ∂A^{act}.

The main disadvantage of this approach is that the graph might become unfavourably large for two reasons. First, if the graph contains a sequence of nodes having a very slow fastest route, this route is also contained in the current F^t even if there is a much faster clone equivalent path, see Figure 4.7 for an example. This may cause the inclusion of time steps well beyond any interest for the optimisation task at hand. Second, if there is a



(a) Fastest route graph F^t.



(b) G^{cur} as generated by the approach described in Section 4.4.

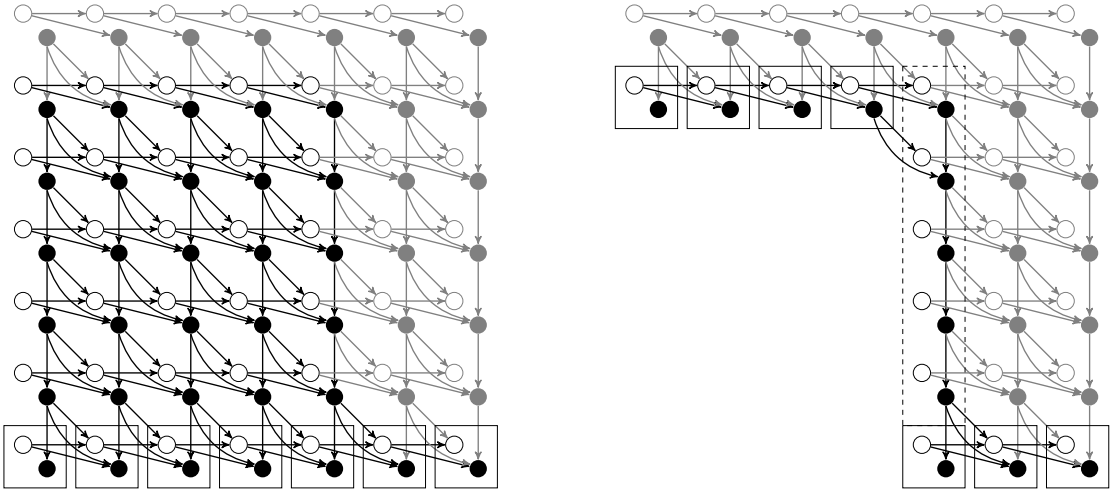
Figure 4.7: Fastest route graph. This picture shows the valid subnetwork generated by the fastest route approach compared with another one that would be generated by the approach described in Section 4.4. The grey nodes are contained in G^{act}, the nodes within the rectangles form F^t resp. G^{cor}. Because the fastest route along the wait nodes (black) is much slower in each step than the fastest route along the run nodes (white), F^t becomes larger each step.

long, possibly enforced, waiting period at one or several clone nodes as in Figure 4.8a, one would prefer the corridor to follow the boundary of G^{act} rather closely, like in Figure 4.8b, in order to reduce the size of the subgraph G^{cur}. In doing so we have to ensure that the three important properties still hold for the improved corridor. This will be considerably more complex than in the simple case above.

4.4 An Improved Corridor for Reducing the Size of Valid Subnetworks

In this section we describe an approach for an improved corridor that tries to follow the "frontier" of G^{act} as closely as possible in order to keep the size of the final G^{cur} small. We will show that under reasonable assumptions the resulting current graph G^{cur} will be only slightly larger than the active graph G^{act}.

We want to mimic the current subnetwork of the grid graph $G^T_{L,n}$ of Example 4.23 and extend it to the general case. The basic idea is to exploit the presence of wait arcs in intermediate stations. The paths that have been constructed in the F^t, $t \in T$, above that are used to construct replacing paths for reducible paths, are always paths along the



(a) Corridor generated by the fastest route approach F^t. This corridor also corresponds to the basic, *unshifted* corridor described in Section 4.4.

(b) "Shifting up" the basic corridor using some wait arcs (dashed rectangle) leads to an improved corridor, which follows the boundary of G^{act} quite closely.

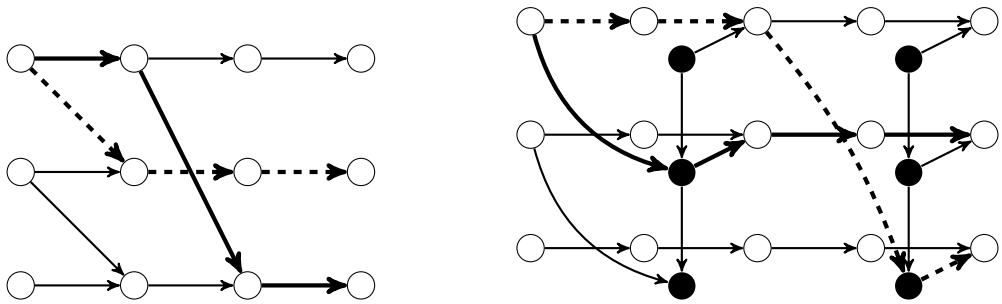
Figure 4.8: Simple and improved corridor. If the active subgraph (grey nodes) contains a long waiting period, a simple fastest route corridor may be quite far away from the active subgraph. In order to reduce the size of the subnetwork to be stored, the corridor may be moved closer to the boundary of the active subgraph if some wait nodes and arcs allow to preserve the corridor properties.

fastest routes. But in general, if F^t is far from G^{act}, then a path in F^t may be replaced by an earlier path as well. A part of a path P in F^t could be "shifted up" to earlier time steps using some intermediate wait nodes, see Figure 4.8 for an illustration.

Although Example 4.23 illustrates the goal of the construction quite well, it does not show the arising difficulties. General problems differ in several aspects from the example grid graph. First, the set of traversal times for $d(uv)$ for some transfer arc $uv \in A$, $u \neq v$, may contain more than one possible traversal time, *i. e.* $|d(e)| > 1$. Second, the structural property (C) allows to compare two paths if they share the same *route*, *i. e.* the same sequence of clone nodes but not necessarily the same sequence of nodes. This property is *not* exploited in the simple fastest path approach using the F^t, $t \in \mathbb{N}_0$, above. In the case of the TTP one can easily imagine that the fastest paths along some route use only run nodes, which model passing through a station without stopping, instead of wait nodes, which mean that the train stops in a station. In order to keep the corridor close to the boundary of G^{act} we will need to use replacing paths along the fastest possible sequence of nodes. A third aspect, which is handled in the F^t as well but not visible in Example 4.23, is the possibility that $[G]$ is not a path but a general acyclic graph.

These structural properties, having $|d(e)| > 1$ and paths along the same routes (*i. e.* $P \neq Q \subset G$ with $[P] = [Q]$), allow the existence of *incomparable* paths that cross each other but have no common intermediate node (no such paths are possible in the grid-graph

example, see Figure 4.9). The existence of such paths that may "jump" over other paths has several consequences that must be respected in the construction of the corridor.



(a) Multiple transfer times. (b) More than one node in a clone node.

Figure 4.9: Crossing paths. If more than one transfer time between two nodes or more than one node in some clone nodes exist, then the time expansion may possess crossing paths (the solid and the dashed path cross).

The construction of the corridor consists of several steps. First we create for each clone node $[u] \in [V]$ a set of *interception nodes* $V^{[u]}$. Each set $V^{[u]}$ is a template of small subgraphs of $[u] \times \mathbb{Z}$ consisting of nodes (u, δ_u) where δ_u is to be interpreted as a relative time offset to a global time shift $t_{[u]} \in T$ applied to $V^{[u]}$ resulting in a shifted copy $V^{[u]}_{t_{[u]}}$. The interception nodes will be constructed so that chaining them together along the fastest possible continuations $\underline{d}(uv)$ yields a valid corridor that fulfils the interception and the continuation property and also the reentrant property (by the same arguments as for the F^t that no path can outrun a fastest path). In fact this corridor will be quite similar to F^t presented in Section 4.3 but exploits the structure of the clone nodes. This basic "fastest" corridor is close to the active subgraph G^{act} if the previously chosen paths do not stop and wait at any clone node. But if active paths include long waiting periods, this basic corridor may be quite far away from G^{act}. So in a second step we improve the corridor by finding better time shifts for the interception nodes $V^{[u]}$ that include waiting possibilities so as to stay as close as possible to G^{act} in order to reduce the size of the subgraph G^{cur}. The concrete structure of the intercepting node sets depends only on the structure of G^T (hence on G and d) and not on the concrete cost function or the structure (and size) of the active subgraph G^{act}. This ensures the important property, that they can be constructed in a preprocessing phase. We will see that by constructing the $V^{[u]}$ properly, it is not difficult to find time-steps that ensure the interception and the continuation property. The reentrant property, however, will *not* be fulfilled automatically and will require the addition of some further elements in the end.

The rest of this section is divided into several paragraphs, each describing one of the construction steps. In Section 4.4.1 we explain the construction of the interception node sets $V^{[u]}$ and show that they can indeed be used to construct a corridor along the fastest routes. We also show that the size of these sets is bounded. Afterwards, in Section 4.4.2, adapted time shifts are determined that shift the interception nodes closer to G^{act} while preserving the continuation and interception properties. We will see that this basic

construction is not sufficient to ensure the reentrant property, so Section 4.4.3 identifies *irreducible nodes* that may lie on reentering paths and must be added to the corridor. Finally, in Section 4.4.4 we put all these together to formulate our final algorithm that constructs an improved corridor G^{cur}. We will show that under reasonable assumptions this corridor exceeds the active subgraph G^{act} at most by a small number of nodes depending only on the structure of G and the transition times d but not on the actual graph G^{act}.

4.4.1 Interception Nodes

In order to facilitate the handling of arc times like $\min \bigcup_{u' \in [u], v' \in [v]} d(u'v')$ for some typical situations, we will use the following notation throughout,

$$\begin{aligned} \underline{d}(u,v) &:= \underline{d}(uv) = \min d(uv), & \bar{d}(u,v) &:= \bar{d}(uv) = \max d(uv) \\ \underline{d}(u,[v]) &:= \min\{\underline{d}(u,v') \colon v' \in [v]\}, & \bar{d}(u,[v]) &:= \max\{\bar{d}(u,v') \colon v' \in [v]\}, \\ \underline{d}([u],v) &:= \min\{\underline{d}(u',v) \colon u' \in [u]\}, & \bar{d}([u],v) &:= \max\{\bar{d}(u',v) \colon u' \in [u]\}, \end{aligned}$$

$$\begin{aligned} \underline{d}([u],[v]) &:= \underline{d}([u][v]) = \min\{\underline{d}(u'v') \colon u' \in [u], v' \in [v]\}, \\ \bar{d}([u],[v]) &:= \bar{d}([u][v]) = \max\{\bar{d}(u'v') \colon u' \in [u], v' \in [v]\}. \end{aligned}$$

Likewise it will be convenient to denote the minimal and maximal time-step of any finite node-like set $X \subset V \times \mathbb{Z}$ by

$\underline{t}(X)$
$$\underline{t}(X) := \begin{cases} \infty, & \text{if } X = \emptyset, \\ \min\{t \colon (x,t) \in X\}, & \text{otherwise,} \end{cases}$$

and

$\bar{t}(X)$
$$\bar{t}(X) := \begin{cases} 0, & \text{if } X = \emptyset, \\ \max\{t \colon (x,t) \in X\}, & \text{otherwise.} \end{cases}$$

We start with the formal description of the requirements on the sets of interception nodes. In the definition of the $V^{[u]}$ below, the terms $\delta_{u'} - \delta_{v'} + \underline{d}([u],[v])$ specify relative time offsets w. r. t. a basic mutual time shift of $\underline{d}([u],[v])$ between $V^{[u]}$ and $V^{[v]}$.

Definition 4.31 A collection of sets $V^{[u]} \subset [u] \times \mathbb{Z}$, $[u] \in [V]$, is called *interception sets* if the following properties are satisfied,

(H1) *(base nodes property)*
$V^{[\hat{u}]} = [\hat{u}] \times \{0\}$ and $[v] \times \{0\} \subseteq V^{[v]}$ for all $[v] \in [V]$,

(H2) *(interception property)* for all $[u][v] \in [A]$,
for $(u', \delta_{u'}) \in [u] \times \mathbb{Z}_-$ and $(v', \delta_{v'}) \in [v] \times \mathbb{N}$ so that $\delta_{v'} - \delta_{u'} + \underline{d}([u],[v]) \in d(u'v')$,
there is a $(v'', \delta_{v''}) \in V^{[v]}$ with $\delta_{v''} - \delta_{u'} + \underline{d}([u],[v]) \in d(u'v'')$ and $\delta_{v''} \leq \delta_{v'}$,

(H3) *(reinterception property)* for all $[u][v] \in [A]$,
for $(u', \delta_{u'}) \in [u] \times \mathbb{N}$ and $(v', \delta_{v'}) \in [v] \times \mathbb{N}$ so that $\delta_{v'} - \delta_{u'} + \underline{d}([u],[v]) \in d(u'v')$ with $\delta_{v'} \leq \bar{t}(V^{[v]})$,
there is a $(v'', \delta_{v''}) \in V^{[v]}$ with $\delta_{v''} - \delta_{u'} + \underline{d}([u],[v]) \in d(u'v'')$ and $\delta_{v''} \leq \delta_{v'}$,

(H4) *(continuation property)* for all $[u][v] \in [A]$,
for $(u', \delta_{u'}) \in V^{[u]}$ there is a $(v', \delta_{v'}) \in V^{[v]}$ with $\delta_{v'} - \delta_{u'} + \underline{d}([u],[v]) \in d(u'v')$.

For $t \in T$ we define the *shifted interception sets*

$$V_t^{[u]} := \left\{ (u, t + \delta_u) \colon (u, \delta_u) \in V^{[u]} \right\}.$$

Although they have rather technical definitions, these conditions are motivated by the considerations above, namely finding a starting node for the replacing path (properties (H1) and (H2)), a continuation within the corridor (property (H4)) and an end node for reconnecting the replacing path (partially by (H3), although we will later see that this is not yet sufficient). We will discuss the consequences of these properties next. For this, assume $[u][v] \in [A]$ and that G^{cor} contains the two sets $V_{t_{[u]}}^{[u]}$, $V_{t_{[v]}}^{[v]}$ with $t_{[v]} - t_{[u]} = \underline{d}([u],[v])$. Figure 4.10 shows the corridor for three clone nodes.

(H1) The *base node property* (Figure 4.11) guarantees that if a path P contains a node at time $t_{[u]}$, it can be intercepted at exactly this node. This property is important to intercept paths that cross the corridor via wait-arcs.

(H2) The *interception property* (Figure 4.12) takes care of paths P that use an arc $(u', t_{u'})(v', t_{v'}) \in A^T(P)$ with $u' \in [u]$, $v' \in [v]$, $t_{u'} \leq t_{[u]}$ and $t_{v'} > t_{[v]}$, *i. e.*, it starts before the time shift $t_{[u]}$ of $V^{[u]}$ but ends after the time shift $t_{[v]}$ of $V^{[v]}$ thereby "jumping" over the initial time line of the corridor. If P then continues with the next node outside G^{cur}, property (H2) allows to use the arc $(u', t_{u'})(v'', t_{v''})$ with $(v'', t_{v''}) \in V_{t_{[v]}}^{[v]}$ in order to redirect the path into the corridor without using any later nodes in $[v]$. Starting from $(v'', t_{v''})$ the replacing path of P can be constructed using (H4).

(H3) The *reinterception property* (Figure 4.13) helps to intercept paths that run, in part, beyond the time line given by the $t_{[u_i]}$ (within or outside of G^{cur}) and then touch G^{cur} in a node v' later than $t_{[v']}$ that is added due to the closure operation, before leaving G^{cur} in the next node. By redirecting the path from v' to $v'' \in V_{t_{[v']}}^{[v']}$ it visits $[v']$ no later and can be continued within G^{cur} by (H4).

(H4) The *continuation property* (Figure 4.14) ensures that each path ending in $(u', t_{u'}) \in V_{t_{[u]}}^{[u]}$ can be continued via an arc $(u', t_{u'})(v', t_{v'})$ with $(v', t_{v'}) \in V_{t_{[v]}}^{[v]}$, thus being continued in the corridor.

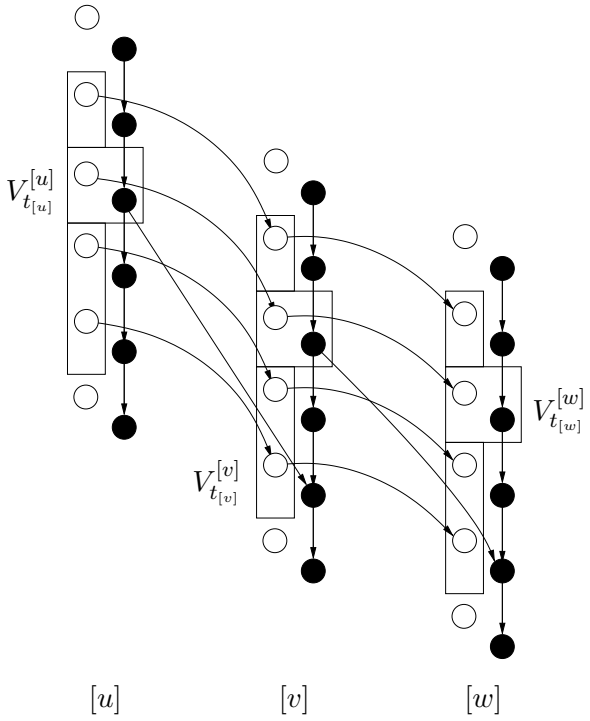


Figure 4.10: Basic corridor. The nodes within the rectangles form the set $V^{[i]}_{t_{[i]}}$, $i \in \{u, v, w\}$ which contains *all* nodes of $[i]$ at time $t_{[i]}$ (small square) and usually only fastest nodes at the other time steps.

The intercepting node sets can be constructed inductively. In order to present one concrete algorithmic possibility we need the following notion.

Definition 4.32 Let $[u][v] \in [A]$, and $(u', \delta_{u'}) \in [u] \times \mathbb{Z}$. The *canonical successor* $N^{[v]}(u', \delta_{u'})$ of $(u', \delta_{u'})$ is a node $(v', \delta_{v'}) \in [v] \times \mathbb{Z}$ with $\delta_{v'} - \delta_{u'} + \underline{d}([u], [v]) = \underline{d}(u', [v])$.

The canonical successor is therefore a fastest continuation from a certain node $(u', \delta_{u'})$ to $[v]$. A mapping $N^{[v]} \colon ([u] \times \mathbb{Z}) \to ([v] \times \mathbb{Z})$ satisfying the definition above is easily provided and is considered as given input data. Based on this, Algorithm 4.2 constructs the interception sets in topological order starting from $\hat{u}$ by adding further nodes to the sets until all properties are fulfilled. A simple but important fact is that all sets $V^{[u]}$, $[u] \in [V]$, remain bounded.

Observation 4.33 *The sets $V^{[u]}$ constructed by Algorithm 4.2 are finite interception sets.*

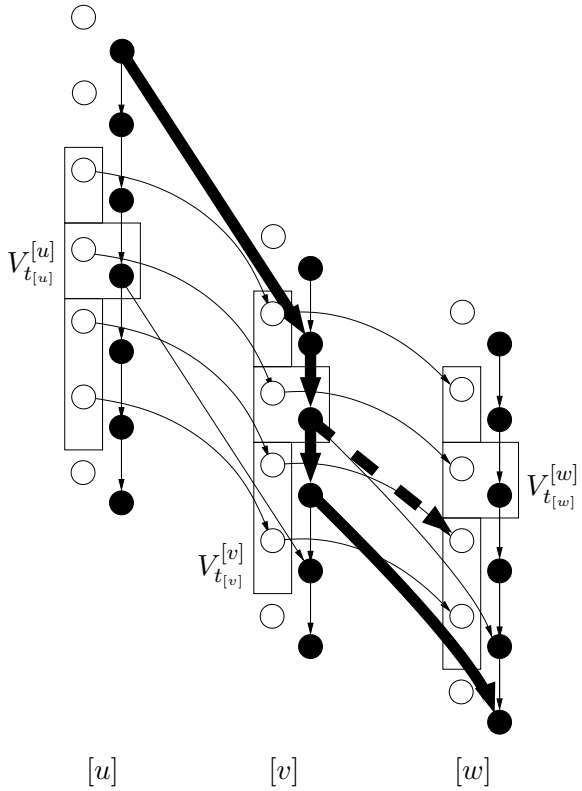


Figure 4.11: Property (H1). The thick path crosses the corridor using wait-arcs and can be intercepted at a node from $[v] \times t_{[v]} \subseteq V^{[v]}_{t_{[v]}}$.

Proof. While (H1) and (H4) can be verified directly, (H2) and (H3) hold because $N^{[v]}(u', \delta_{u'})$ provides the node $(v'', \delta_{v''})$ with smallest possible $\delta_{v''}$. Finiteness follows because $[G]$ and all sets $d(a)$, $a \in A$, are finite. □

Note that in general these sets may become quite large if the time sets $d(a)$, $a \in A$, have a complex structure. In practice, most instances have well structured running-times and the sets $V^{[u]}$ remain relatively small. The following observation gives an example for this under the reasonable assumption that there is a canonical "best" node $n^{[v]}$ for all $[v] \in [V]$ which is used by fastest paths in G^T.

Observation 4.34 *Suppose for all* $[v] \in [V]$ *there is a node* $n^{[v]} \in [v]$ *so that*

$$\forall\, [u][v] \in [A], \forall\, (u', \delta_{u'}) \in [u] \times \mathbb{Z} \colon N^{[v]}(u', \delta_{u'}) \in \{n^{[v]}\} \times \mathbb{Z}, \tag{4.6}$$

and

$$\forall\, [v][w] \in [A] \colon \min d(n^{[v]}, n^{[w]}) = \underline{d}([v], [w]). \tag{4.7}$$

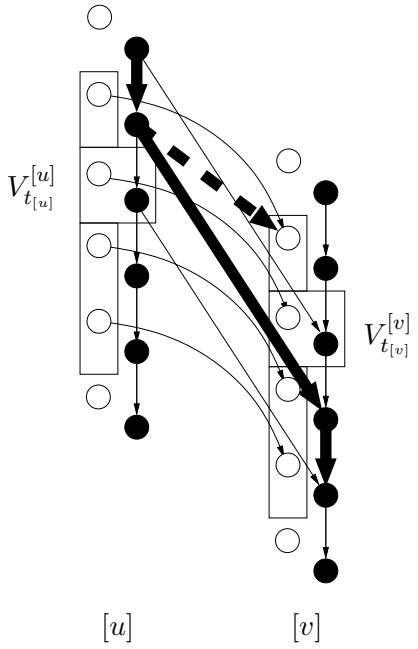


Figure 4.12: Property (H2). The thick path jumping over the corridor using an arc $(u, t_u)(v, t_v), t_u < t_{[u]}, t_v > t_{[v]}$ can be redirected using the dashed arc into $V^{[v]}_{t_{[v]}}$.

Then the sets $V^{[v]}$, $[v] \in [V]$, as constructed by Algorithm 4.2 fulfil

$$V^{[v]} \subseteq ([v] \times \{0\}) \cup (\{n^{[v]}\} \times \{-\bar{d}, \dots, \bar{d}\})$$

where

$$\bar{d} := \max\{\bar{d}(uv) \colon uv \in A\}.$$

Proof. The claim is certainly true for $V^{[\hat{u}]}$. So let $[v] \in [V]$ and assume as induction hypothesis the claim holds for all $V^{[u]}$ with $[u][v] \in [A]$. Let $(v', \delta_{v'}) \in V^{[v]}$. First we observe

$$\text{for all } (u', \delta_{u'}) \in [u] \times \mathbb{Z} \text{ with } (v', \delta_{v'}) = N^{[v]}(u', \delta_{u'}) \text{ there holds } \delta_{u'} \leq \delta_{v'}, \tag{4.8}$$

which is clear by definition of $N^{[v]}(u', \delta_{u'})$, because $\delta_{v'} - \delta_{u'} + \underline{d}([u], [v]) = \underline{d}(u', [v]) \geq \underline{d}([u], [v])$. If $(v', \delta_{v'}) \in ([v] \times \{0\})$, the claim is clear. Otherwise $(v', \delta_{v'}) \in W_i^{[v]}$ for some $i \in \{2, 3, 4\}$ (see Algorithm 4.2) which implies $v' = n^{[v]}$ by (4.6). We consider three cases:

1. If $(v', \delta_{v'}) \in W_2^{[v]}$, then there must be a $(u', \delta_{u'}) \in [u] \times \mathbb{Z}_-$ and a $(\tilde{v}, \delta_{\tilde{v}}) \in [v] \times \mathbb{N}$

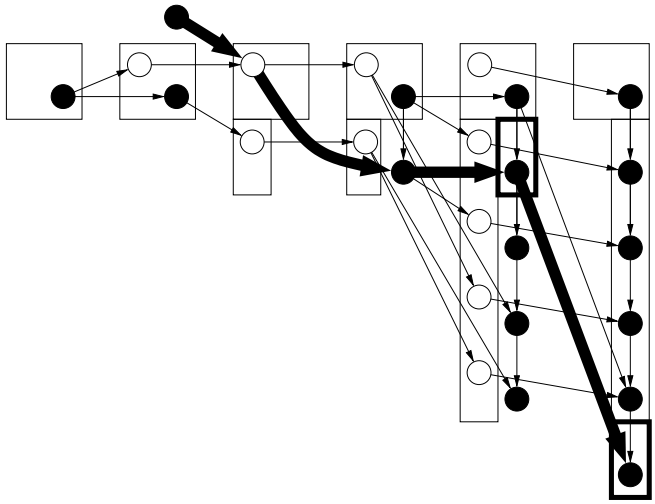


Figure 4.13: Property (H3). The thick path cannot be intercepted and replaced by an earlier path within the corridor, because it is crossed by all potential replacing paths. Thus the node in the thick rectangle in $[u_4]$ must be added to $V^{[u_4]}_{t_{[u_4]}}$. Consequently (because of (H4)) the node in the thick rectangle in $[\check{u}]$ must be added to $V^{[\check{u}]}_{t_{[\check{u}]}}$, too.

with $\delta_{\tilde{v}} - \delta_{u'} + \underline{d}([u],[v]) \in d(u'\tilde{v})$ and $\delta_{v'} \leq \delta_{\tilde{v}}$. Because $\delta_{\tilde{v}} > 0$ we have

$$\underbrace{\delta_{\tilde{v}}}_{>0} - \delta_{u'} + \underbrace{\underline{d}([u],[v])}_{\geq 0} \leq \bar{d}, \text{ hence } \delta_{u'} > -\bar{d},$$

which implies by (4.8)

$$\delta_{v'} \geq \delta_{u'} > -\bar{d},$$

and similarly

$$\delta_{v'} - \underbrace{\delta_{u'}}_{\leq 0} + \underline{d}([u],[v]) \leq \bar{d}, \text{ thus } \delta_{v'} \leq \bar{d}.$$

2. If $(v', \delta_{v'}) \in W_4^{[v]}$, then there is a $(u', \delta_{u'}) \in V^{[u]}$ for some $[u][v] \in [A]$ with $(v', \delta_{v'}) = N^{[v]}(u', \delta_{u'})$. On the one hand the induction hypothesis on $V^{[u]}$ implies $\delta_{v'} \geq \delta_{u'} \geq -\bar{d}$. On the other hand if $u' \neq n^{[u]}$ we have by induction hypothesis $\delta_{u'} = 0$ and therefore

$$\delta_{v'} - \underbrace{\delta_{u'}}_{=0} + \underbrace{\underline{d}([u],[v])}_{\geq 0} \leq \bar{d}, \text{ hence } \delta_{v'} \leq \bar{d}.$$

If $u' = n^{[u]}$, (4.7) implies $\min d(u'n^{[v]}) = \underline{d}([u],[v])$ and therefore

$$\delta_{v'} - \delta_{u'} + \underline{d}([u],[v]) = \underline{d}([u],[v]) \quad \Longleftrightarrow \quad \delta_{v'} = \delta_{u'},$$

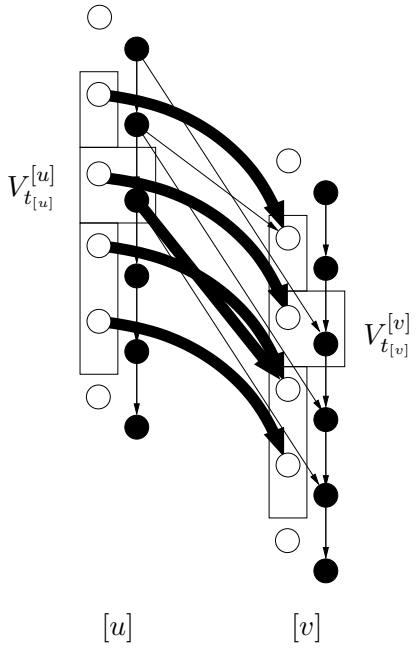


Figure 4.14: Property (H4). Each path leading on to a node in $V^{[u]}_{t_{[u]}}$ can be continued via one of the thick arcs to a node in $V^{[v]}_{t_{[v]}}$.

thus $\delta_{v'} \leq \bar{d}$.

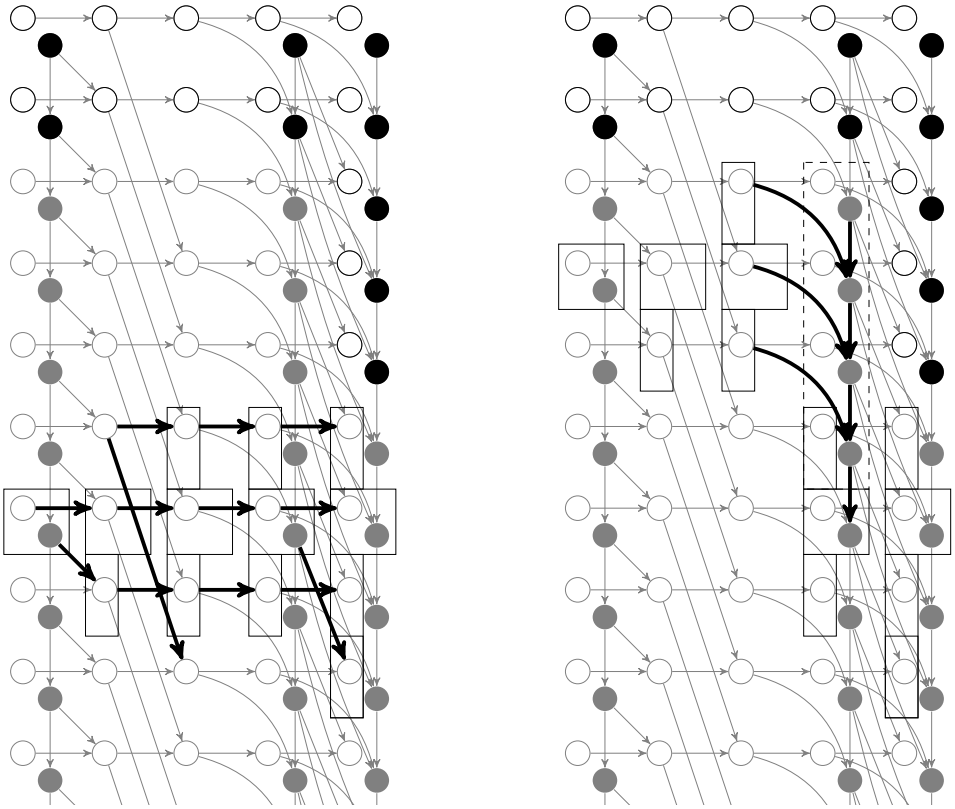
3. If $(v', \delta_{v'}) \in W_3^{[v]}$, then there must be a $(u', \delta_{u'}) \in [u] \times \mathbb{N}$ and a $(\tilde{v}, \delta_{\tilde{v}}) \in [v] \times \mathbb{N}$ with $\delta_{\tilde{v}} - \delta_{u'} + \underline{d}([u], [v]) \in d(u'\tilde{v})$ and $\delta_{\tilde{v}} \leq \bar{t}(W_2^{[v]} \cup W_4^{[v]}) \leq \bar{d}$ (by the previous two cases). Consequently $\delta_{v'} \leq \delta_{\tilde{v}}$ because $(v, \delta_{v'}) \in N^{[v]}(u', \delta_{u'})$. Furthermore, we have $\delta_{u'} > 0$, thus (4.8) implies $-\bar{d} < 0 < \delta_{u'} \leq \delta_{v'}$. □

The corridor that gives the fastest continuation along any clone path corresponds to the case where the concrete time shifts $t_{[u]}$ for the sets $V^{[u]}$ are chosen so that

1. $V^{[u]}_{t_{[u]}} \subset V^T \setminus V^{\text{act}}$ for $[u] \in [V]$ and
2. for $[u][v] \in [A]$ the difference of time shifts is $t_{[v]} - t_{[u]} = \underline{d}([u], [v])$. In this case the properties (H1)–(H4) permit a construction of G^{cor} similar to the fastest route approach of the previous section.

An example of a fastest corridor constructed this way is shown in Figure 4.15a.

Remark In fact, the difference between the fastest route approach of the previous section and the fastest continuation corridor above is caused by the structural properties of the clone nodes. Using clone nodes, it is sufficient to include the fastest paths



(a) Fastest corridor. The interception sets are connected using the fastest possible continuation (always $\min d([u][v]) = 0$ in this example). The thick arcs are the generating arcs according to the construction Algorithm 4.2.

(b) Shifted corridor. The interception sets are moved closer to the active subgraph using wait arcs. This is only possible if the shift is large enough so that replacing arcs using the wait arcs can exist. The thick arcs are the wait arcs used for the continuation.

Figure 4.15: Corridor construction. The black nodes are G^{act}, the corridor is build from the interception sets, which are the nodes contained in the boxes.

Algorithm 4.2: ConstructInterceptionNodes

Input : Graph $G = (V, A)$, traversal time function d, $N^{[u]}$ ($[u] \in [V]$)
Output: Interception nodes $\{V^{[u]} \colon [u] \in [V]\}$
$V^{[\hat{u}]} := [\hat{u}] \times \{0\}$
for $[v] \in [V]$ *with* $V^{[u]}$ *computed for all* $[u][v] \in [A]$ **do**
  $W_1^{[v]} := [v] \times \{0\}$
  $W_4^{[v]} := \{N^{[v]}(u', \delta_{u'}) \colon [u][v] \in [A], (u', \delta_{u'}) \in V^{[u]}\}$
  $W_2^{[v]} := \{N^{[v]}(u', \delta_{u'}) \colon [u][v] \in [A], (u', \delta_{u'}) \in [u] \times \mathbb{Z}_-, \exists\, (v', \delta_{v'}) \in [v] \times \mathbb{N},$
    $\delta_{v'} - \delta_{u'} + \underline{d}([u], [v]) \in d(u'v')\}$
  $W_3^{[v]} := \{N^{[v]}(u', \delta_{u'}) \colon [u][v] \in [A], (u', \delta_{u'}) \in [u] \times \mathbb{N}, \exists\, (v', \delta_{v'}) \in [v] \times \mathbb{N},$
    $\delta_{v'} - \delta_{u'} + \underline{d}([u], [v]) \in d(u'v'), \delta_{v'} \leq \bar{t}(W_2^{[v]} \cup W_4^{[v]})\}$
  $V^{[v]} := W_1^{[v]} \cup W_2^{[v]} \cup W_3^{[v]} \cup W_4^{[v]}$
return $\{V^{[u]} \colon [u] \in [V]\}$

along each sequence of clones instead of each sequence of nodes. However, the fastest continuation corridor is not yet sufficient to form a valid subnetwork, see the discussion of *irreducible paths* below in Section 4.4.3.

4.4.2 Time Shifts

In order to improve the corridor we allow time shifts larger than $\underline{d}([u], [v])$. These must be selected carefully to preserve the continuation and interception properties. In order to preserve the continuation property, larger mutual time offsets than $\underline{d}([u], [v])$ typically require the availability of a waiting node $v' \in [v]$ with $v'v' \in A$ that allows to bridge the time difference. Figure 4.15b illustrates the structure of a shifted corridor when such wait nodes are present. The possibility to use these waiting paths is incorporated below in the extended versions of the properties (H1)–(H4) formulated w. r. t. two sets $V_{t_{[u]}}^{[u]}$ and $V_{t_{[v]}}^{[v]}$ with $[u][v] \in [A]$ and sufficiently large $t_{[u]}, t_{[v]} \in T$. In fact, (H1)–(H4) imply those extended properties if $t_{[v]} - t_{[u]} = \underline{d}([u], [v])$ which ensures the existence of feasible time shifts (we will prove these statements formally below).

Definition 4.35 Let $[u][v] \in [A]$ and let $t_{[u]}, t_{[v]} \in T$ satisfy $V_{t_{[u]}}^{[u]}, V_{t_{[v]}}^{[v]} \subset V^T \setminus V^{\mathrm{act}}$. The two shifted interception sets $V_{t_{[u]}}^{[u]}$ and $V_{t_{[v]}}^{[v]}$ *match for* $[u][v]$ if the following conditions hold

(H1') $[v] \times \{t_{[v]}\} \subseteq V_{t_{[v]}}^{[v]}$,

(H2') *(extended interception property)*
for $(u', t_{u'}) \in [u] \times T$ with $t_{u'} \leq t_{[u]}$ and $(v', t_{v'}) \in [v] \times T$ with $t_{v'} > t_{[v]}$ and $(u', t_{u'})(v', t_{v'}) \in A^T$ there is a path $P' = (u', t_{u'})(v'', t_{v''}) \dots (v'', t'_{v''}) \subset A^T \setminus A^{\text{act}}$ with $(v'', t'_{v''}) \in V^{[v]}_{t_{[v]}}$ and $t_{v''} \leq t_{v'}$,

(H3') *(extended reinterception property)*
for $(u', t_{u'}) \in [u] \times T$ with $t_{u'} > t_{[u]}$ and $(v', t_{v'}) \in [v] \times T$ with $t_{[v]} < t_{v'} \leq \bar{t}(V^{[v]}_{t_{[v]}})$ and $((u', t_{u'}), (v', t_{v'})) \in A^T$ there is a path $P' = (u', t_{u'})(v'', t_{v''}) \dots (v'', t'_{v''}) \subset A^T \setminus A^{\text{act}}$ with $(v'', t'_{v''}) \in V^{[v]}_{t_{[v]}}$ and $t_{v''} \leq t_{v'}$,

(H4') *(extended continuation property)*
for $(u', t_{u'}) \in V^{[u]}_{t_{[u]}}$ there exists a path $P = (u', t_{u'})(v', t_{v'}) \dots (v', t'_{v'}) \subset A^T \setminus A^{\text{act}}$ with $(v', t'_{v'}) \in V^{[v]}_{t_{[v]}}$.

Figure 4.16 illustrates the extended properties.

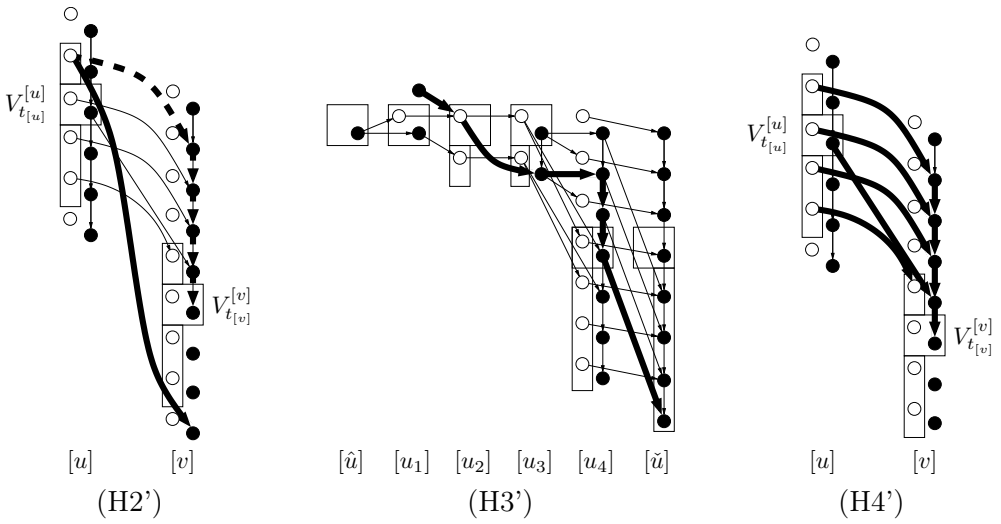


Figure 4.16: Properties (H2')–(H4') are the same as (H2)–(H4) but instead of directly connecting arcs additional waiting arcs may be used as long as the path is contained in $G^T \setminus G^{\text{act}}$.

Observation 4.36 *Let $[u][v] \in [A]$ and $t_{[u]}, t_{[v]} \in T$ such that $V^{[u]}_{t_{[u]}}, V^{[v]}_{t_{[v]}} \subset V^T \setminus V^{act}$ and $t_{[v]} - t_{[u]} = \underline{d}([u], [v])$. Then $V^{[u]}_{t_{[u]}}$ and $V^{[v]}_{t_{[v]}}$ match.*

Proof. Note that for $t_{[v]} - t_{[u]} = \underline{d}([u],[v])$ the condition $(u', \delta_{u'} + t_{[u]})(v', \delta_{v'} + t_{[v]}) \in A^T$ is equivalent to $\delta_{v'} - \delta_{u'} + \underline{d}([u],[v]) \in d(u'v')$. For $V^{[u]}_{t_{[u]}}, V^{[v]}_{t_{[v]}} \subset V^T \setminus V^{\mathrm{act}}$ the definition of shifted interception nodes, $V^{[u]}_{t_{[u]}} = \{(u', \delta_{u'} + t_{[u]})\colon (u', \delta_{u'}) \in V^{[u]}\}$, and properties (H1)–(H4) therefore ensure that each of the paths required by (H1)–(H4) can indeed be realised by a single arc in A^T. This arc cannot be in A^{act} because the corresponding head node is in $V^{[v]}_{t_{[v]}} \subset V^T \setminus V^{\mathrm{act}}$. □

Considering now a clone node with several outgoing clone arcs, it will, in general, not be possible to have just one copy of an interception set so as to connect to all successive interception sets. Therefore, for each clone arc $[u][v] \in [A]$ a matching time shift $t_{[u]}$ will have to be available within a set of time shifts $T^{[u]}$ for each time shift $t_{[v]}$ in the set of time shifts $T^{[v]}$ of $[v]$. If $[u]$ contains a waiting node $u' \in [u]$ with $u'u' \in A$ it may be possible to collapse some of them and it is one goal of the design to keep the sets $T^{[u]}$ small.

Definition 4.37 Let $[u][v] \in [A]$ and let, for $[w] \in \{[u],[v]\}$, $T^{[w]} \subset T$ satisfy $V^{[w]}_t \subset V^T \setminus V^{\mathrm{act}}$ for $t \in T^{[w]}$. The two sets $T^{[u]}$ and $T^{[v]}$ *match*, if for every $t_{[v]} \in T^{[v]}$ there is a matching $t_{[u]} \in T^{[u]}$, *i. e.*, $V^{[u]}_{t_{[u]}}$ and $V^{[v]}_{t_{[v]}}$ match.

A family $\mathfrak{T} = \{T^{[u]} \subset T\}_{[u]\in[V]}$ of nonempty finite sets is called *admissible time shifts (for interception sets $V^{[u]}$, $[u] \in [V]$, and G^{act})* or simply *admissible* if for each $[u][v] \in [A]$ the sets $T^{[u]}$ and $T^{[v]}$ match.

An admissible family $\mathfrak{T}$ is called *reduced* no set $T^{[u]}$ contains unnecessary time shifts, *i. e.*, if there does not exist an admissible family $\tilde{\mathfrak{T}} = \{\tilde{T}^{[u]}\}_{[u]\in[V]}$ with $\tilde{T}^{[u]} \subseteq T^{[u]}$ for all $[u] \in [V]$ and $\tilde{T}^{[\tilde{u}]} \subsetneq T^{[\tilde{u}]}$ for some $[\tilde{u}] \in [V]$.

Note that for $[u][v] \in [A]$ the requirement for matching sets $T^{[u]}$ and $T^{[v]}$ is not symmetric: not every $t \in T^{[u]}$ needs a matching $t' \in T^{[v]}$. The existence of admissible time shifts is not difficult to see but essential.

Observation 4.38 *For any G^{act} there exist admissible time shifts $T^{[u]} \subset T$, $[u] \in [V]$.*

Proof. Define the sets in reverse topological order starting with $T^{[\tilde{u}]} = \{\tau\}$ for some $\tau \in \mathbb{N}$ large enough. For each $[u] \in [V]$ with the sets $T^{[v]}$ already defined for all $[u][v] \in [A]$, put $T^{[u]} = \{t - \underline{d}([u],[v])\colon t \in T^{[v]}, [u][v] \in [A]\}$. By Observation 4.36 and τ large enough this ensures that for each $[u][v] \in [A]$ the sets $T^{[u]}$ and $T^{[v]}$ match. □

Admissible time shifts for G^{act} allow to find matching time shifts $t_{[u_i]}$ along any path $P = [u_1] \dots [u_k] \subset [G]$ starting with an arbitrary $t_{[u_k]} \in T^{[u_k]}$ in the last node of the path.

Proposition 4.39 *Let $\mathcal{T} = \{T^{[u]}\}_{[u]\in[V]}$ be admissible and let $[u_1]\dots[u_k] \subseteq [A]$ be a clone-path. For each $t_{[u_k]} \in T^{[u_k]}$ there exist $t_{[u_i]} \in T^{[u_i]}$, $i = 1,\dots,k-1$, so that $t_{[u_i]}$ and $t_{[u_{i+1}]}$ match. Given any such choice, for all $j, j' \in \{1,\dots,k\}$, $j \le j'$, and all $(v_j, t'_j) \in V^{[u_j]}_{t_{[u_j]}}$ there is a path*

$$P = (v_j, t'_j)(v_{j+1}, t_{j+1})\dots(v_{j+1}, t'_{j+1})\dots(v_{j'}, t'_{j'}) \subset A^T \setminus A^{act}$$

with $(v_i, t'_i) \in V^{[u_i]}_{t_{[u_i]}}$ for all $i \in \{j,\dots,j'\}$.

Proof. Pick any $t_{[u_k]} \in T^{[u_k]}$ and continue recursively for $i = k-1,\dots,1$ by picking a $t_{[u_i]} \in T^{[u_i]}$ so that $V^{[u_i]}_{t_{[u_i]}}$ and $V^{[u_{i+1}]}_{t_{[u_{i+1}]}}$ match (this is possible by definition because the $T^{[u]} \subset T$, $[u] \in [V]$, are admissible). The second claim now follows from applying (H4') iteratively starting from (v_j, t'_j). □

Given admissible time shifts $\mathcal{T}$ for G^{act}, we consider the graph $G^{\mathcal{T}} = (V^{\mathcal{T}}, A^{\mathcal{T}}) = \mathrm{cl}(\bigcup_{[u]\in[V]} V^{[u]}_{\max T^{[u]}})$. The following technical proposition justifies the first part of our construction, namely, each path leaving $G^{\mathcal{T}}$ without returning to $G^{\mathcal{T}}$ can be intercepted and replaced by a path staying in $G^{\mathcal{T}} \setminus G^{\mathrm{act}}$.

$G^{\mathcal{T}} = (V^{\mathcal{T}}, A^{\mathcal{T}})$

Proposition 4.40 *Given admissible time shifts $\mathcal{T} = \{T^{[u]}\}_{[u]\in[V]}$ for interception sets $V^{[u]}$, $[u] \in [V]$, and G^{act}, let $G^{\mathcal{T}} = \mathrm{cl}(\bigcup_{[u]\in[V]} V^{[u]}_{\max T^{[u]}})$ and*

$$P = (u_1, t_1)\dots(u_1, t'_1)(u_2, t_2)\dots(u_2, t'_2)\dots(u_k, t'_k) \subset A^T \setminus A^{act}$$

be a path with $k > 1$, $[u_i] \neq [u_j]$ for all $1 \le i < j \le k$, $(u_1, t_1) \in \partial A^{act}$ and $t'_k > t_{[u_k]}$ for some $t_{[u_k]} \in T^{[u_k]}$. Then there is a path $P' = (u_1, t_1)P' \le_T P$ with $P' \subset A^T \setminus A^{act}$ having its last node in $V^{[u_k]}_{t_{[u_k]}}$.

Proof. Let $t_{[u_1]},\dots,t_{[u_k]}$ denote time shifts associated with the clone-path P of $[P]$ as defined by Proposition 4.39. Let $(u_i, t) \in V^T(P)$ be the node with

$$t = \min\left\{t'_i, \bar{t}\left(V^{[u_i]}_{t_{[u_i]}}\right)\right\} \quad \text{and} \quad t_j > \bar{t}\left(V^{[u_j]}_{t_{[u_j]}}\right) \text{ for } i < j \le k$$

(this node exists because $t_1 \le t_{[u_1]} \le \bar{t}(V^{[u_1]}_{t_{[u_1]}})$). Depending on this time t we have to consider three cases to construct the replacing path P'.

1. If $t_i \le t < t_{[u_i]}$ then $(u_i, t) \neq (u_k, t'_k)$. Because $t+1 \le t_{[u_i]} \le \bar{t}(V^{[u_i]}_{t_{[u_i]}})$ the choice of t implies $t = t'_i$ and $((u_i, t), (u_{i+1}, t_{i+1})) \in P$ with $t_{i+1} > \bar{t}(V^{[u_{i+1}]}_{t_{[u_{i+1}]}}) \ge t_{[u_{i+1}]}$. Now property (H2') implies the existence of a path $Q_1 = (u_i, t) \dots (\tilde{u}_{i+1}, \tilde{t}_{i+1}) \subset A^T \backslash A^{\text{act}}$ with $(\tilde{u}_{i+1}, \tilde{t}_{i+1}) \in V^{[u_{i+1}]}_{t_{[u_{i+1}]}}$ and $\tilde{t}_{i+1} < t_{i+1}$. By Proposition 4.39 there is a path $Q_2 = (\tilde{u}_{i+1}, \tilde{t}_{i+1}) \dots (\tilde{u}_k, \tilde{t}_k) \subset A^T \setminus A^{\text{act}}$ with $Q_2 \le_T (u_{i+1}, t'_{i+1})P$. Connecting these parts together we get a path $P' = P(u_i, t)Q_1(\tilde{u}_{i+1}, \tilde{t}_{i+1})Q_2(\tilde{u}_k, \tilde{t}_k)$ with the property $P' \le_T P$.

2. If $t_i \le t_{[u_i]} \le t$ then $(u_i, t_{[u_i]}) \in V^T(P)$ and (H1') implies $(u_i, t_{[u_i]}) \in V^{[u_i]}_{[t_{u_i}]}$. If $[u_i] = [u_k]$ then $P' = P(u_i, t_{[u_i]})$ satisfies the requirements. Otherwise $[u_i] \neq [u_k]$ and we use Proposition 4.39 to get a path $Q = (u_i, t_{[u_i]}) \dots (\tilde{u}_k, \tilde{t}_k) \subset A^T \setminus A^{\text{act}}$ with $(\tilde{u}_k, \tilde{t}_k) \in V^{[u_k]}_{[t_{u_k}]}$. The choice of t ensures $Q \le_T (u_i, t_{[u_i]})P(u_k, t'_k)$. The path $P' := P(u_i, t_{[u_i]})Q(\tilde{u}_k, \tilde{t}_k)$ has the desired property.

3. If $t_{[u_i]} < t_i \le t$ we know by assumption that $(u_i, t_i) \neq (u_1, t_1)$. Depending on whether the time step t'_{i-1} of the predecessor node (u_{i-1}, t'_{i-1}) is not greater or greater than $t_{[u_{i-1}]}$ either (H2') or (H3') ensure that there is a path $Q_1 := (u_{i-1}, t'_{i-1}) \dots (\tilde{u}_i, \tilde{t}_i) \subset A^T \setminus A^{\text{act}}$ with $(\tilde{u}_i, \tilde{t}_i) \in V^{[u_i]}_{t_{[u_i]}}$ and $\tilde{t}_i \le t_i$. Using Proposition 4.39 there is a path $Q_2 = (\tilde{u}_i, \tilde{t}_i) \dots (\tilde{u}_k, \tilde{t}_k)$ with $Q_2 \le_T (u_i, t_i)P$ and the path
$$P' := P(u_{i-1}, t'_{i-1})Q_1(\tilde{u}_i, \tilde{t}_i)Q_2(\tilde{u}_k, \tilde{t}_k)$$
has the desired property. □

4.4.3 Reentering Paths

As mentioned before, the node sets $\bigcup\{V^{[u]}_{t_{[u]}}\}$ are the main building part of V^{cor} for a path $[P]$, but this is not sufficient by itself. Indeed, if we do not include the fastest paths along all node sequences, it may well happen that paths reenter the graph $G^{\mathfrak{T}}$, in particular in regions where the difference in time shifts is big. Rigorously, a ∂A^{act}-path P is called *reentering* if it leaves $G^{\mathfrak{T}}$, *i. e.*, $V^T(P) \cap (V^T \setminus V^{\mathfrak{T}}) \neq \emptyset$, and later returns to $V^{\mathfrak{T}}$. For such a path the following situation might arise. We would like to replace the part of P that is not contained in $G^{\mathfrak{T}}$ by a path P'. Assume (w, t_w) is the first node of P when P *returns* to $G^{\mathfrak{T}}$. Then we require $P'(w', t'_w) \le_T P(w, t_w)$ with $w' \in [w]$. Because the remaining part of P, namely $(w, t_w)P$, may be contained in G^{act} it cannot be replaced in general. But this implies that the replacing path P' has to meet P in (w, t_w) while satisfying $P' \le_T P$, *i. e.* $(w, t_w) = (w', t_{w'})$. In some cases this is simply impossible within $G^{\mathfrak{T}}$ (see Figure 4.17 for an example).

The last step of the construction of G^{cur} therefore adds some further nodes and arcs to $G^{\mathfrak{T}}$ in those situations where connections as described above may be missing. In fact, they can conveniently be described as those nodes of G^{act}-irreducible paths that are not contained in $V^{\mathfrak{T}}$.

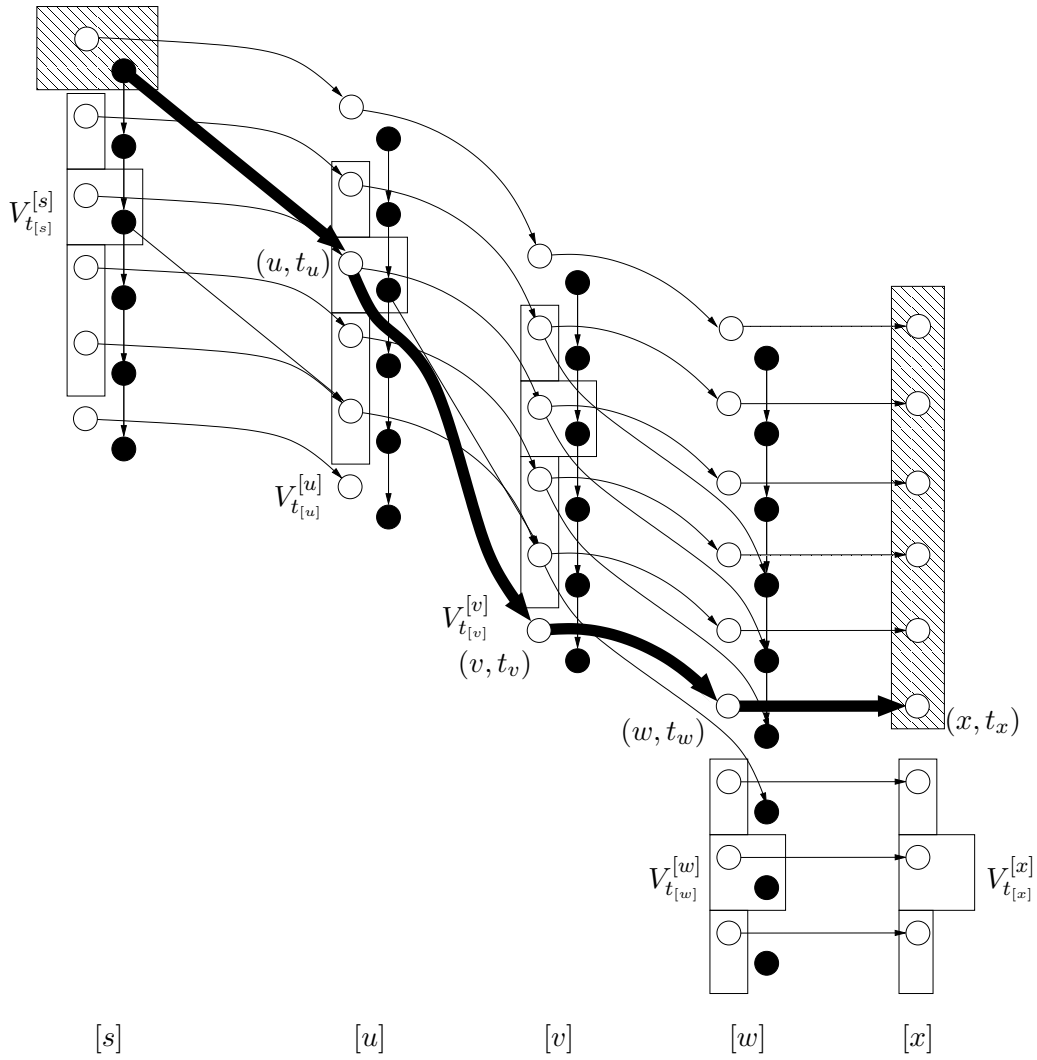


Figure 4.17: Reentering paths. The hatched region denotes G^{act}. The thick ∂A^{act}-path $P = (s, t_s)(u, t_u)(v, t_v)(w, t_w)(x, t_x)$ has no replacing path because there is no path from (u, t_u) to (w, t_w) within $V^{[u]}_{t_{[u]}}$, $V^{[v]}_{t_{[v]}}$ and $V^{[w]}_{t_{[w]}}$. Therefore the node (v, t_v) has to be included in G^{cur}.

Definition 4.41 Let $\mathcal{T} = \{T^{[u]}\}_{[u]\in[V]}$ be admissible and put $G^{\mathcal{T}} = \mathrm{cl}(\bigcup_{[u]\in[V]} V^{[u]}_{\max T^{[u]}}, \emptyset)$. A node $(u,t_u) \in V^T \setminus V^{\mathcal{T}}$ is *irreducible (for $G^{\mathcal{T}}$)* if it is contained in an G^{act}-irreducible path. The set of all irreducible nodes is denoted by $B^{\mathcal{T}}$.

Observation 4.42 *The last node of any G^{act}-irreducible path is contained in $G^{\mathcal{T}}$. In particular,*

$$\bar{t}(B^{\mathcal{T}} \cap ([\check{u}] \times T)) \leq \bar{t}(V^{\mathcal{T}} \cap ([\check{u}] \times T)) = \bar{t}(V^{[\check{u}]}_{\max T^{[\check{u}]}}). \tag{4.9}$$

Proof. Let $P = (u,t_u)P(v,t_v)$ be an G^{act}-irreducible path, and so by definition a $\partial A^{\mathrm{act}}$-path, and assume $(v,t_v) \notin G^{\mathcal{T}}$, hence $t_v > \bar{t}(V^{[v]}_{\max T^{[v]}})$. Together with $\bar{t}(V^{\mathrm{act}} \cap ([u]\times T)) < \bar{t}(V^{[u]}_{\max T^{[u]}})$ for each $[u] \in [V]$, this implies $v = \check{u}$. Now use Proposition 4.40 with respect to the path P to find a path $Q = (u,t_u)Q(\check{u},t)$ with $Q \leq_T P$, $Q \subset A^T \setminus A^{\mathrm{act}}$ and $t \leq \bar{t}(V^{[\check{u}]}_{\max T^{[\check{u}]}})$. This Q is a $\partial A^{\mathrm{act}}$-path and $Q <_T P$, so P is reducible, a contradiction.□

This shows that the set of irreducible nodes is finite. To obtain a valid subnetwork it suffices to add all irreducible nodes to $G^{\mathcal{T}}$, because then all G^{act}-irreducible paths are included in this graph.

Corollary 4.43 *The set of irreducible nodes satisfies $B^{\mathcal{T}} \subseteq [V] \times \{1,\dots,\bar{t}(V^{\mathcal{T}})\}$ and $\mathrm{cl}(V^{\mathcal{T}} \cup B^{\mathcal{T}})$ is a valid subnetwork whenever (C) holds.*

Proof. By Observation 4.42 any node $(u,t_u) \in B^{\mathcal{T}}$ lies on an irreducible path $P(v,t_v)$ for some end node $(v,t_v) \in V^{\mathcal{T}}$. Because $\underline{d}([u],[v]) \geq 0$, any predecessor $(u',t_{u'})$ of (v,t_v) on P must satisfy $t_{u'} \leq t_v$, so $t_u \leq t_v \leq \bar{t}(V^{\mathcal{T}})$. Because $\mathrm{cl}(V^{\mathcal{T}} \cup B^{\mathcal{T}})$ is finite and contains all G^{act}-irreducible paths, the result follows by Corollary 4.28. □

Given upper bounds on the time indexes of irreducible nodes belonging to successor clone nodes, a rough bound on the irreducible nodes of the clone node itself is readily deduced via the usual fastest continuation argument.

Observation 4.44 *Let $[u] \in [V] \setminus \{[\check{u}]\}$ be a clone node and $\mathcal{T} = \{T^{[v]}\}_{[v]\in[V]}$ an admissible family of time shifts. Assume that for each node $[v] \in [V]$ with $[u][v] \in [A]$ we are given a bound*

$$\bar{b}_{[v]} \geq \bar{t}((V^{\mathcal{T}} \cup B^{\mathcal{T}}) \cap ([v] \times T)).$$

Then it holds for any node $(u', t_{u'}) \in [u] \times T$ that lies on a G^{act}-irreducible path using an edge $(u', t_{u'})(v, t_v)$ with $u' \in [u] \neq [v]$

$$t_{u'} \leq \bar{b}_{[v]} - \underline{d}([u], [v]). \tag{4.10}$$

In particular, if $\mathcal{T}$ is reduced,

$$\bar{t}((V^{\mathcal{T}} \cup B^{\mathcal{T}}) \cap ([u] \times T)) \leq \max\left\{\bar{b}_{[v]} - \underline{d}([u], [v])\colon [u][v] \in [A]\right\}.$$

Proof. Let $(u', t_{u'}) \in [u] \times T$ and P be an G^{act}-irreducible path containing an arc $(u', t_{u'})(v, t_v) \in P$, $[u'] \neq [v]$. By definition of $B^{\mathcal{T}}$ the node (v, t_v) must be either contained in $V^{\mathcal{T}}$ or it is irreducible and therefore contained in $B^{\mathcal{T}}$. Consequently $t_v \leq \bar{b}^{[v]}$ and because $t_v - t_{u'} \in d(u', v)$ we have

$$t_{u'} \leq t_v - \underline{d}(u', v) \leq \bar{b}_{[v]} - \underline{d}([u], [v]).$$

For the last inequality first choose $t_{[u]} \in T^{[u]}$ arbitrarily and a node $(u', t_{u'}) \in V^{[u]}_{t_{[u]}}$. Because $\mathcal{T}$ is admissible and reduced there must be an arc $[u][v] \in [A]$ and a time shift $t_{[v]} \in T^{[v]}$ so that $V^{[u]}_{t_{[u]}}$ and $V^{[v]}_{t_{[v]}}$ match, and a path $P = (u', t_{u'})(v, t_v) \dots (v, t'_v)$ with $(v, t'_v) \in V^{[v]}_{t_{[v]}}$ by (H4'). This proves

$$\bar{t}(V^{\mathcal{T}} \cap ([u] \times T)) \leq \max\left\{\bar{b}_{[v]} - \underline{d}([u], [v])\colon [u][v] \in [A]\right\}.$$

Now assume $(u', t_{u'}) \in B^{\mathcal{T}} \cap ([u] \times T)$ with $t_{u'} > \bar{t}(V^{\mathcal{T}} \cap ([u] \times T))$ and let P be a G^{act}-irreducible path containing $(u', t_{u'})$. By Observation 4.42 the end node of P must be contained in $V^{\mathcal{T}}$ and because $G^{\mathcal{T}}$ is closed and by the choice of $t_{u'}$ this end node cannot be in $[u] \times T$. Therefore P must be of the form

$$P = P(u', t_{u'}) \dots (u', t'_{u'})(v, t_v)P$$

with $[u][v] \in [A]$. Relation (4.10) implies $t'_{u'} \leq \bar{b}_{[v]} - \underline{d}([u], [v])$ and with $t_{u'} \leq t'_{u'}$ we conclude

$$\bar{t}(B^{\mathcal{T}} \cap ([u] \times T)) \leq \max\left\{\bar{b}_{[v]} - \underline{d}([u], [v])\colon [u][v] \in [A]\right\}. \qquad \square$$

The next result presents a feasible interval of admissible larger time shifts with respect to a successor clone node that contains at least one waiting node. These larger time shifts allow to bring $G^{\mathcal{T}}$ closer to G^{act} but do not yet help to reduce the bound on the time indexes of the irreducible nodes.

Proposition 4.45 *Given a node $v \in V$ with $(v,v) \in A$, a time shift $t_{[v]} \in T$ with $V^{[v]}_{t_{[v]}} \subset V^T \setminus V^{act}$ and an arc $[u][v] \in [A]$, put*

$$\underline{\delta}_{[u]} := \min\{\delta_{u'} \colon (u', \delta_{u'}) \in V^{[u]}\} \quad (\leq 0),$$
$$\bar{\delta}_{[u]} := \max\{\delta_{u'} \colon (u', \delta_{u'}) \in V^{[u]}\} \quad (\geq 0),$$
$$\underline{\tau}^v_{[u]} := \max\left\{1 + \bar{t}(V^{act} \cap ([u] \times T)), \bar{t}(V^{act} \cap ([v] \times T)) - \underline{d}([u], v)\right\} - \underline{\delta}_{[u]}, \tag{4.11}$$
$$\bar{d}^v_{[u]} := \max\left\{\bar{d}([u],[v]), \bar{\delta}_{[u]} + \max_{u' \in [u]} \underline{d}(u', v)\right\}. \tag{4.12}$$

For any t with

$$\underline{\tau}^v_{[u]} \leq t \leq t_{[v]} - \bar{d}^v_{[u]} \tag{4.13}$$

the sets $V^{[u]}_t$ and $V^{[v]}_{t_{[v]}}$ match.

Proof. Suppose t satisfies (4.13). Note that for $(u', t_{u'}) = (u', t + \delta_{u'}) \in V^{[u]}_t$ with $(u', \delta_{u'}) \in V^{[u]}$ we have

$$t_{u'} \overset{(4.13)}{\geq} \underline{\tau}^v_{[u]} + \underline{\delta}_{[u]} \overset{(4.11)}{\geq} 1 + \bar{t}(V^{\text{act}} \cap ([u] \times T)),$$

so $V^{[u]}_t \cap V^{\text{act}} = \emptyset$ as required in Definition 4.35. Condition (H1') is satisfied, because $V^{[v]}$ satisfies (H1). To see that condition (H4') holds, observe that any node $(u', t_{u'}) \in V^{[u]}_t$ (in fact, any node $(u', t_{u'})$ with $t_{u'} \in [t + \underline{\delta}_{[u]}, t + \bar{\delta}_{[u]}]$) has an arc $(u', t_{u'})(v, t_v) \in A^T \setminus A^{\text{act}}$ with $t_v - t_{u'} \in [\underline{d}([u], v), \max_{u'' \in [u]} \underline{d}(u'', v)]$, so

$$t_v \geq t_{u'} + \underline{d}([u], v) \geq \underline{\tau}^v_{[u]} + \underline{\delta}_{[u]} + \underline{d}([u], v) \overset{(4.11)}{\geq} \bar{t}(V^{\text{act}} \cap ([v] \times T))$$

and

$$t_v \leq t_{u'} + \max_{u'' \in [u]} \underline{d}(u'', v) \leq t + \bar{\delta}_{[u]} + \max_{u'' \in [u]} \underline{d}(u'', v) \overset{(4.12)}{\leq} t + \bar{d}^v_{[u]} \overset{(4.13)}{\leq} t_{[v]}.$$

Therefore this arc can be continued by a path $(v, t_v) \dots (v, t_{[v]}) \subset A^T \setminus A^{\text{act}}$ with $(v, t_{[v]}) \in V^{[v]}_{t_{[v]}}$ by (H1'). This proves (H4') and, by the same line of arguments, (H3'), as well. Because $t_{[v]} - t \overset{(4.13)}{\geq} \bar{d}^v_{[u]} \overset{(4.12)}{\geq} \bar{d}([u],[v])$, there can be no arcs that give rise to requirements in (H2') and so (H2') holds. This shows that $V^{[u]}_t$ and $V^{[v]}_{t_{[v]}}$ match. □

One possibility to obtain better bounds on the time indexes of irreducible nodes is to show that beyond a certain time index all nodes must be part of G^{act}-reducible paths.

The corridor $G^{\mathcal{T}}$ offers good possibilities to design earlier paths that lead up to the same clone node at which the path in question is still outside $G^{\mathcal{T}}$. If it is possible to link into the original path again coming from a slightly earlier node within this clone node, the path is shown to be reducible. In order to do this only on basis of local information for all larger time steps, we need the possibility to wait at a node that offers a sufficiently slow connection to the next node on the path. The precise requirements in view of several possible clone node successors are collected in the next result.

Proposition 4.46 *Let $\mathcal{T} = \{T^{[u]}\}_{[u]\in[V]}$ be admissible time shifts and $[u][v] \in [A]$, $[v] \neq [\check{u}]$, be an arc such that the following properties hold:*

(i) There exists a $t_{[u]} \in T^{[u]}$ so that for all $v' \in [v]$ with $v'v' \in A$ and for all $(u', t_{u'}) \in V^{[u]}_{t_{[u]}}$ there is an arc $(u', t_{u'})(v', t_{v'}) \in A^T$ with $t_{v'} \geq \bar{t}(V^{act} \cap ([v] \times T))$.

(ii) For each $[v][w] \in [A]$ there is a node $v_{[w]} \in [v]$ with $v_{[w]}v_{[w]} \in A$ so that $\bar{d}(v_{[w]}, w') \geq \bar{d}([v], w')$ for all $w' \in [w]$.

Then there is no G^{act}-irreducible path containing an arc $(u', t_{u'})(v', t_{v'})$ with $u' \in [u]$, $v' \in [v]$ and

$$t_{u'} > \max\left\{\bar{t}(V^{[u]}_{t_{[u]}}),\, \vec{t}^{\,t_{[u]}}_{[u][v]} + \max_{[v][w]\in[A]}\{\bar{d}(v_{[w]}, [w]) - \underline{d}([v], [w])\} - \underline{d}([u], [v])\right\} \tag{4.14}$$

where

$$\vec{t}^{\,t_{[u]}}_{[u][v]} := \max_{\substack{v'\in[v],\\ v'v'\in A}} \max_{(u',t_{u'})\in V^{[u]}_{t_{[u]}}} \min\{t_{v'} \geq \bar{t}(V^{act} \cap ([v] \times T))\colon (u', t_{u'})(v', t_{v'}) \in A^T\}. \tag{4.15}$$

Proof. Given $[u][v] \in [A]$ so that (i) and (ii) hold, let P be a $\partial A^{\mathrm{act}}$-path containing an arc $(u', t_{u'})(v', t_{v'})$ with $u' \in [u]$, $v' \in [v]$ and $t_{u'}$ satisfying (4.14). Then

$$\begin{aligned} t_{v'} &\geq t_{u'} + \underline{d}([u], [v]) \\ &> \vec{t}^{\,t_{[u]}}_{[u][v]} + \max_{[v][w]\in[A]}\{\bar{d}(v_{[w]}, [w]) - \underline{d}([v], [w])\} \\ &\geq \bar{t}(V^{\mathrm{act}} \cap ([v] \times T)). \end{aligned} \tag{4.16}$$

Use Proposition 4.40 with respect to the path $P(u', t_{u'})$ and $t_{[u]}$ to find a path $Q <_T P(u', t_{u'})$ in $A^T \setminus A^{\mathrm{act}}$ ending in a node $(u'', t_{u''}) \in V^{[u]}_{t_{[u]}}$ ($t_{u''} < t_{u'}$ by (4.14)).

First consider the case $v'v' \in A$. By (i) and (4.15) there is an arc $(u'', t_{u''})(v', t^0_{v'}) \in A^T$ with $\bar{t}(V^{\mathrm{act}} \cap ([v] \times T)) \leq t^0_{v'} \leq \vec{t}^{\,t_{[u]}}_{[u][v]}$ and by (4.16) $t_{v'} > \vec{t}^{\,t_{[u]}}_{[u][v]}$, thus there is a path $Q' = (u'', t_{u''})(v', t^0_{v'}) \dots (v', t_{v'}) \subset A^T \setminus A^{\mathrm{act}}$. Hence, $Q(u'', t_{u''})Q'(v', t_{v'})P <_T P$ and P is reducible.

So we may assume $v'v' \notin A$. Note that $(v', t_{v'}) \notin V^{\text{act}}$ because $t_{v'} > \bar{t}(V^{\text{act}} \cap ([v] \times T))$ by (4.16). Thus, it has a successor $(w', t_{w'})$ in P with $[w'] \neq [v]$ and

$$\begin{aligned} t_{w'} - \bar{d}(v_{[w']}, w') &\geq t_{v'} + \underline{d}([v], [w']) - \bar{d}(v_{[w']}, w') \\ &\geq t_{v'} - \max_{[v][w] \in [A]} \{\bar{d}(v_{[w]}, [w]) - \underline{d}([v], [w])\} \overset{(4.16)}{>} \bar{t}^{t_{[u]}}_{[u][v]}. \end{aligned}$$

In consequence and using assumption (ii) there is an arc $(v_{[w']}, t'_{v_{[w']}})(w', t_{w'}) \in A^T$ with $t_{w'} - t'_{v_{[w']}} = \bar{d}(v_{[w']}, w')$, thus $\bar{t}^{t_{[u]}}_{[u][v]} < t'_{v_{[w']}} \leq t_{v'}$. Exploiting (i) as in the first case this allows to find a path $Q' = (u'', t_{u''})(v_{[w']}, t_{v_{[w']}}) \dots (v_{[w']}, t'_{v_{[w']}}) \subset A^T \setminus A^{\text{act}}$ with $Q' <_T (u', t'_u)P(v', t'_v)$. Now $Q(u'', t_{u''})Q'(v_{[w']}, t'_{v_{[w']}})(w', t_{w'})P <_T P$, so P is reducible.□

The requirements of the result seem rather stringent but are quite natural in practice where there is typically just one waiting node in each clone node, and where starting from a waiting position takes longer than continuing in full speed. Of course, the improved bound along $[u][v]$ is helpful in $[u]$ only if there is no other successor of $[u]$ giving rise to a worse bound.

4.4.4 The Algorithm

Putting the constructions of the interception sets $V^{[u]}$, $[u] \in [V]$, the admissible time shifts $T^{[u]}$, $[u] \in [V]$, and the bounds on the irreducible nodes together, we are now ready to state an algorithm that computes a corridor G^{cur} and the subgraph $G^{\mathfrak{T}}$. An algorithmic possibility to construct admissible time shifts and upper bounds on the time indexes of irreducible nodes is described in Algorithm 4.3 (it uses the notations introduced in Proposition 4.45 and Proposition 4.46). In order to improve readability some parts of the algorithm are extracted as subroutines in Algorithms 4.4 to 4.6. In the following we will use the notation A$n.l$ to refer to line number l in algorithm n.

Remark 4.47

1. The more involved quantities in these algorithms do not depend on G^{act} and can be precomputed (*e. g.*, $\bar{d}^{[v]}_{[u]}$, $\Delta_{[u][v]}$ and the validity of (ii) of Proposition 4.46 in A4.6.4 are constants), or can be computed much simpler than the corresponding definitions suggest (e.g., in Algorithm 4.6, $\bar{t}(V^{[u]}_{\hat{t}^{[u][v]}}) = \hat{t}^{[u][v]} + \bar{\delta}_{[u]}$ and for the given choices, $\bar{t}^{\hat{t}^{[u][v]}}_{[u][v]} = \hat{t}^{[u][v]} + \max_{v' \in [v], (v', v') \in A} \max_{(u', \delta_{u'}) \in V^{[u]}} (\delta_{u'} + \underline{d}(u', v'))$ by (4.19) below) that we avoided so as to highlight the connection to the original objects. In fact, the algorithm turns out to be very efficiently implementable (see Observation 4.48 below).
2. In many situations the $T^{[u_i]}$ and $\bar{t}^{[u_i]}$ computed by Algorithm 4.3 will not yield the smallest possible graph G^{cur}. Indeed, several conditions could be stated or computed in more sophisticated manner. The current choices are meant to keep

Algorithm 4.3: ConstructCorridor

Input : Interception node sets $V^{[u]}$, $[u] \in [V]$
Output: Valid subnetwork G^{cur}
// Compute lower bounds on time shifts
$\{\underline{t}_{[u]} \colon [u] \in [V]\} \leftarrow$ ComputeLowerLimits(G^{act})
// Construct time shifts and upper time limits in reverse topological order
1 $T^{[\check{u}]} := \{\underline{t}_{[\check{u}]}\}$, $\bar{t}^{[\check{u}]} := \bar{t}(V^{[\check{u}]}_{\max T^{[\check{u}]}})$
for $[u] \in [V] \setminus \{[\check{u}]\}$ *with* $T^{[v]}$, $\bar{t}^{[v]}$ *computed for all* $[u][v] \in [A]$ **do**
  $T^{[u]} \leftarrow \emptyset$, $\bar{t}^{[u]} \leftarrow \bar{t}(V^{\text{act}} \cap ([u] \times T))$
  for $[u][v] \in [A]$ **do**
    if $[v]$ *contains no node* v' *with* $v'v' \in A$ **then**
2 $(T^{[u][v]}, \bar{t}^{[u][v]}) \leftarrow$ ComputeNowaitTimeshifts$\big([u][v], T^{[v]}, \bar{t}^{[v]}\big)$
    else
3 $(T^{[u][v]}, \bar{t}^{[u][v]}) \leftarrow$ ComputeWaitTimeshifts$\big([u][v], \underline{t}^{[u]}, T^{[v]}, \bar{t}^{[v]}\big)$
4 $T^{[u]} \leftarrow T^{[u]} \cup T^{[u][v]}$
5 $\bar{t}^{[u]} \leftarrow \max\left\{\bar{t}^{[u]}, \bar{t}^{[u][v]}, \bar{t}(V^{[u]}_{\max T^{[u][v]}})\right\}$
return $G^{cur} := \text{cl}\left(\bigcup_{[u]\in[V]}([u] \times \bar{t}^{[u]}), \emptyset\right)$

the presentation somewhat manageable without significant loss in performance for most practical situations.

Observation 4.48 *The update of the* $T^{[u]}$ *and* $\bar{t}^{[u]}$ *for a new* G^{act} *requires*

$$O\left(|[A]| \cdot \max_{[u]\in[U]} |T[u]|\right)$$

steps and only depends on the size and structure of the clone graph $[G]$*. In particular, if* $[G]$ *is a path, then* $\max_{[u]\in[U]} |T^{[u]}| = 1$ *and the running time is linear in the number of clone nodes.*

Proof. In view of Remark 4.47 the running times of the subroutines are $O(|[A]|)$ for Algorithm 4.4, and $O(|T^{[v]}|)$ for Algorithms 4.5 and 4.6. Therefore the running time of Algorithm 4.3 is bounded by $O(|[A]| \cdot \max\{|T^{[u]}| \colon [u] \in [V]\})$. If $[G]$ is a path then $|T^{[\check{u}]}| = 1$ by A4.3.1 and because $\forall\, [u] \in [V] \setminus \{\check{u}\} \colon |\{[u][v] \in [A]\}| = 1$ the set $T^{[u][v]}$ returned by either Algorithm 4.5 or Algorithm 4.6 contains exactly one time step. □

Algorithm 4.4: ComputeLowerLimits

Input : Active subgraph $G^{\mathrm{act}} = (V^{\mathrm{act}}, A^{\mathrm{act}})$
Output: Lower limits for time shifts $\{\underline{t}_{[u]} \colon [u] \in [V]\}$
$\underline{t}_{[\hat{u}]} := 1 + \bar{t}(V^{\mathrm{act}} \cap ([\hat{u}] \times T))$
for $[v] \in [V] \setminus \{[\hat{u}]\}$ *with* $\underline{t}_{[u]}$ *computed for all* $[u][v] \in [A]$ **do**
1 $\quad \underline{t}_{[v]} := \max\left(\left\{\underline{t}_{[u]} + \underline{d}([u],[v]) \colon [u][v] \in [A]\right\} \cup \left\{1 + \bar{t}(V^{\mathrm{act}} \cap ([v] \times T)) - \underline{\delta}_{[v]}\right\}\right)$
return $\{\underline{t}_{[u]} \colon [u] \in [V]\}$

Algorithm 4.5: ComputeNowaitTimeshifts

Input : Arc $[u][v]$, time shifts $T^{[v]}$, upper time limit $\bar{t}^{[v]}$
Output: Time shifts $T^{[u][v]}$, upper time limit $\bar{t}^{[u][v]}$
$T^{[u][v]} := \{t - \underline{d}([u],[v]) \colon t \in T^{[v]}\}$
$\bar{t}^{[u][v]} := \bar{t}^{[v]} - \underline{d}([u],[v])$
return $(T^{[u][v]}, \bar{t}^{[u][v]})$

We are now ready to state the main result of this chapter.

Theorem 4.49 *For any G^{act} and interception sets $V^{[u]}$, $[u] \in [V]$, Algorithm 4.3 computes a valid subnetwork $G^{cur} = \mathrm{cl}\left(\bigcup_{[u]\in[V]} [u] \times \bar{t}^{[u]}\right)$ whenever (C) holds.*

Proof. We first show that $\mathcal{T} = \{T^{[u]}\}_{[u]\in[V]}$ is admissible. In order to see for every $[u] \in [V]$ and $t \in T^{[u]}$ that $V_t^{[u]} \cap V^{\mathrm{act}} = \emptyset$, it suffices to prove $\min T^{[u]} \geq \underline{t}_{[u]}$, because $\underline{t}_{[u]} \geq 1 + \bar{t}(V^{\mathrm{act}} \cap ([u] \times T)) - \underline{\delta}_{[u]}$ by A4.4.1. For $[\check{u}]$ this holds because $T^{[\check{u}]} = \{\underline{t}_{[\check{u}]}\}$ by A4.3.1. By induction in reverse topological order the first term in the max of A4.4.1 implies $\underline{t}_{[v]} \geq \underline{t}_{[u]} + \underline{d}([u],[v])$ for each arc $[u][v] \in [A]$, thus $t - \underline{d}([u],[v]) \geq \underline{t}_{[u]}$ for all $t \in T^{[v]}$. Therefore if for an arc $[u][v] \in [A]$, $T^{[u][v]}$ is computed by Algorithm 4.5 then $\min T^{[u][v]} \geq \underline{t}_{[u]}$. If otherwise $T^{[u][v]}$ is computed by Algorithm 4.6 in A4.6.3 then together with A4.6.1 there holds $\min T^{[u][v]} \geq \underline{t}_{[u]}$, too. The update of $T^{[u]}$ in A4.3.4 leads to $\min T^{[u]} \geq \underline{t}_{[u]}$.

Next we show by induction in reverse topological order of the nodes $[u]$ that for any $[u][v] \in [A]$ each $t_{[v]} \in T^{[v]}$ has a matching $t_{[u]} \in T^{[u]}$. For $[\check{u}]$ there is nothing to show, so let $[u] \in [V]$, $[u] \neq [\check{u}]$. When Algorithm 4.5 or Algorithm 4.6 is called for $[u][v]$ then the computation of $T^{[v]}$ is already complete. By Observation 4.36 Algorithm 4.5 puts a matching $t_{[u]}$ into $T^{[u][v]}$ (and hence finally in $T^{[u]}$) for each $t_{[v]} \in T^{[v]}$. Algorithm 4.6 does the same for all $t_{[v]} \in T^{[v]}$ with $t_{[v]} < t_{[u][v]}$, so it remains to show that $\underline{\tau}_{[u][v]}$ matches all $t_{[v]} \in T^{[v]}$ with $t_{[v]} \geq t_{[u][v]}$. For this, observe that by A4.6.1 and (4.11) as well as

Algorithm 4.6: ComputeWaitTimeshifts

Input : Arc $[u][v]$, lower time limit $\underline{t}^{[u]}$, time shifts $T^{[v]}$, upper time limit $\bar{t}^{[v]}$
Output: Time shifts $T^{[u][v]}$, upper time limit $\bar{t}^{[u][v]}$

`// Compute a common time shift connectable to all` $v' \in [v]$ `with` $(v', v') \in A$

1 $\underline{\tau}_{[u][v]} := \max \left\{ \underline{t}_{[u]}, \bar{t}(V^{\text{act}} \cap ([v] \times T)) - \min\limits_{\substack{v' \in [v], \\ (v', v') \in A}} \underline{d}([u], v') - \underline{\delta}_{[u]} \right\}$

2 $\bar{d}^{[v]}_{[u]} := \max \left\{ \bar{d}([u], [v]), \bar{\delta}_{[u]} + \max\limits_{\substack{v' \in [v], \\ (v', v') \in A}} \max\limits_{u' \in [u]} \underline{d}(u', v') \right\}$

$t_{[u][v]} := \underline{\tau}_{[u][v]} + \bar{d}^{[v]}_{[u]}$

`// Ensure a matching time shift for each` $t \in T^{[v]}$

3 $T^{[u][v]} := \{\underline{\tau}_{[u][v]} \colon t_{[u][v]} \leq t \in T^{[v]}\} \cup \{t - \underline{d}([u], [v]) \colon t_{[u][v]} > t \in T^{[v]}\}$

`// Compute upper time limit`

4 **if** $\max T^{[u][v]} \geq \underline{\tau}_{[u][v]}$ *and* $[u][v]$ *satisfies condition (ii) of Proposition 4.46* **then**

`// Determine an upper bound on the time index of irreducible nodes`

5 $\hat{t}^{[u][v]} := \min\{t \in T^{[u][v]} \colon t \geq \underline{\tau}_{[u][v]}\}$

6 $\Delta_{[u][v]} := \max\left\{\bar{d}(v_{[w]}, [w]) - \underline{d}([v], [w]) \colon [v][w] \in [A]\right\} - \underline{d}([u], [v])$

7 $\bar{\tau}^{[u][v]} := \max\left\{\bar{t}(V^{[u]}_{\hat{t}^{[u][v]}}), \bar{t}^{\hat{t}^{[u][v]}}_{[u][v]} + \Delta_{[u][v]}\right\}$

8 $\bar{t}^{[u][v]} := \min\{\bar{t}^{[v]} - \underline{d}([u], [v]), \bar{\tau}^{[u][v]}\}$

else

9 $\bar{t}^{[u][v]} := \bar{t}^{[v]} - \underline{d}([u], [v])$

return $(T^{[u][v]}, \bar{t}^{[u][v]})$

A4.6.2 and (4.12)

$$\underline{\tau}_{[u][v]} \geq \max_{\substack{v' \in [v], \\ (v', v') \in A}} \underline{\tau}^{v'}_{[u]} \quad \text{and} \quad \bar{d}^{[v]}_{[u]} = \max_{\substack{v' \in [v], \\ (v', v') \in A}} \bar{d}^{v'}_{[u]}.$$

Thus, the condition $t_{[v]} \geq t_{[u][v]} = \underline{\tau}_{[u][v]} + \bar{d}^{[v]}_{[u]}$ implies

$$\max_{\substack{v' \in [v], \\ (v', v') \in A}} \underline{\tau}^{v'}_{[u]} \leq \underline{\tau}_{[u][v]} \leq t_{[v]} - \max_{\substack{v' \in [v], \\ (v', v') \in A}} \bar{d}^{v'}_{[u]}.$$

Therefore Proposition 4.45 proves that $t_{[u]} = \underline{\tau}_{[u][v]}$ and $t_{[v]}$ match in this case. This establishes that $\mathcal{T}$ constructed by Algorithm 4.3 is admissible.

It remains to show that $\bar{t}^{[u]} \geq \bar{t}((V^{\mathcal{T}} \cup B^{\mathcal{T}}) \cap ([u] \times T))$ holds for all $[u] \in [V]$, because then G^{cur} contains all G^{act}-irreducible paths and is therefore a valid subnetwork given (C) by Corollary 4.28. The claim holds for $[\check{u}]$ by A4.3.1 and Observation 4.42. Once

more using induction in reverse topological order let $[u] \in [V]$, $[u] \neq [\check{u}]$, and suppose that $\bar{t}^{[v]} \geq \bar{t}((V^{\mathcal{T}} \cup B^{\mathcal{T}}) \cap ([v] \times T))$ holds for all $[u][v] \in [A]$. By Observation 4.44, for each $[u][v] \in [A]$ any node $(u', t_{u'})$ with $u' \in [u]$ lying on a G^{act}-irreducible path passing a node of $[v]$ must satisfy $t_{u'} \leq \bar{t}^{[v]} - \underline{d}([u], [v])$. This proves the bound for all irreducible nodes lying on an irreducible path leading on to nodes $[v]$ that are treated either in Algorithm 4.5 or A4.6.9. If, for some $[v]$, $\bar{t}^{[u]}$ is determined by $\bar{t}^{[u][v]}$ computed within A4.6.8 then $\underline{\tau}_{[u][v]} \leq \hat{t}^{[u][v]} \in T^{[u][v]} \subseteq T^{[u]}$ by A4.6.4 and A4.6.5. Now observe that in A4.6.8 $[v]$ satisfies requirement (ii) of Proposition 4.46 and $t_{[u]} = \hat{t}^{[u][v]}$ guarantees that for all $v' \in [v]$ with $(v', v') \in A$ and all $(u', t_{u'}) \in V^{[u]}_{\hat{t}^{[u][v]}}$ there is an arc $(u', t_{u'})(v', t_{v'}) \in A^T$ with $t_{v'} \geq \bar{t}(V^{\mathrm{act}} \cap ([v] \times T))$. Indeed, for the arc $(u', t_{u'})(v', t_{v'}) \in A^T$ with $t_{v'} - t_{u'} = \underline{d}(u', v')$ we obtain

$$t_{v'} \geq t_{u'} + \underline{d}([u], v') \overset{(u', t_{u'}) \in V^{[u]}_{\hat{t}^{[u][v]}}}{\geq} \hat{t}^{[u][v]} + \underline{\delta}_{[u]} + \underline{d}([u], v')$$
$$\overset{\hat{t}^{[u][v]} \geq \underline{\tau}_{[u][v]}}{\geq} \underline{\tau}_{[u][v]} + \underline{\delta}_{[u]} + \underline{d}([u], v') \overset{A4.6.1}{\geq} \bar{t}(V^{\mathrm{act}} \cap ([v] \times T)).$$

Because the definition A4.6.7 of $\bar{\tau}^{[u][v]}$ just matches or exceeds the right-hand side of (4.14), Proposition 4.46 asserts the validity of the bound $\bar{\tau}^{[u][v]}$ for irreducible paths using an edge from $[u]$ to $[v]$. Finally, whenever $\bar{t}^{[u]}$ is updated in A4.3.5 for any $[v]$ it is guaranteed that $\bar{t}^{[u]} \geq \bar{t}(V^{[u]}_{\max T^{[u][v]}})$, thus $\bar{t}^{[u]} \geq \bar{t}((V^{\mathcal{T}} \cup B^{\mathcal{T}}) \cap ([u] \times T))$ holds for all $[u] \in [V]$, completing the proof. □

In the case of general routing graphs it is difficult to obtain reasonable bounds on the number of time steps included on top of G^{act}, because a clone node that is not needed at all within G^{act} may be required to be included with high time indexes in $G^{\mathcal{T}}$, because it has a potential predecessor having high time indexes on G^{act}. Likewise, the necessity to include fastest paths along all, in particular also along the slowest clone route, may already induce a high lower bound $\underline{t}_{[u]}$ far off from $\bar{t}(V^{\mathrm{act}} \cap ([u] \times T))$.

Useful bounds seem only to be obtainable when $[G]$ is restricted to a path, which is actually the case in our practical application of Chapter 2. A specialised algorithm for path graphs $[G]$ would exploit that all time shift sets $T^{[u]}$ hold a single time value $t_{[u]}$, but we refrain from this here, because we would like to highlight in what way the given Algorithm 4.3 improves on the construction of Theorem 4.30 already in the case of $[G]$ being a clone path. In general, closeness to G^{act} requires the availability of clone nodes with waiting nodes. We first show for every clone arc $[u][v]$ that has a $v' \in [v]$ with $v'v' \in A$, that the time shifts of $T^{[u]}$ matching a time shift of $T^{[v]}$ are within a constant of $\bar{t}(V^{\mathrm{act}} \cap ([v] \times T))$.

Observation 4.50 *Suppose that $[G] = [\hat{u} = u_1][u_2] \dots [\check{u} = u_k]$ is a path and that Algorithm 4.3 executes Algorithm 4.6 for an arc $[u][v] \in [A]$. Then*

$$\max T^{[u]} \leq 1 + \bar{t}(V^{act} \cap ([v] \times T)) + \bar{d}([u], [v]) - 2\underline{d}([u], [v]) + \bar{\delta}_{[u]} - \underline{\delta}$$

where $\underline{\delta} = \min\{\underline{\delta}_{[u_i]} : i = 1, \dots, k\}$ with $\underline{\delta}_{[u]}$ and $\bar{\delta}_{[u]}$ defined in Proposition 4.45.

Proof. By induction on step 1 of Algorithm 4.4 there holds for $i = 2, \ldots, k$

$$\underline{t}_{[u_i]} \leq 1 + \bar{t}(V^{\text{act}} \cap ([u_i] \times T)) - \underline{\delta},$$

because $\bar{t}(V^{\text{act}} \cap ([u_i] \times T)) \geq \bar{t}(V^{\text{act}} \cap ([u_{i-1}] \times T)) + \underline{d}([u_{i-1}], [u_i])$ due to $[G]$ being a path. Thus, by A4.6.1 and $\underline{\delta} \leq 0$, the value $\underline{\tau}_{[u][v]}$ may be bounded by

$$\underline{\tau}_{[u][v]} \leq 1 + \bar{t}(V^{\text{act}} \cap ([v] \times T)) - \underline{\delta} - \underline{d}([u], [v]), \tag{4.17}$$

and from A4.6.2 we obtain $\bar{d}^{[v]}_{[u]} \leq \bar{d}([u], [v]) + \bar{\delta}_{[u]}$. Therefore, in A4.6.3 a $t_{[v]} \in T^{[v]}$ with $t_{[v]} < t_{[u][v]} = \underline{\tau}_{[u][v]} + \bar{d}^{[v]}_{[u]}$ gives rise to a time shift

$$t_{[u]} = t_{[v]} - \underline{d}([u], [v]) \leq 1 + \bar{t}(V^{\text{act}} \cap ([v] \times T)) - \underline{\delta} - 2\underline{d}([u], [v]) + \bar{d}([u], [v]) + \bar{\delta}_{[u]}.$$

Because this bound is greater than the upper bound on $\underline{\tau}_{[u][v]}$, this completes the proof.□

In the same way, a bound can be derived for the value $\bar{t}^{[u]}$ whenever Algorithm 4.6 reaches line 8.

Observation 4.51 *Suppose that $[G] = [\hat{u} = u_1][u_2] \ldots [\check{u} = u_k]$ is a path and that Algorithm 4.3 executes line 8 in Algorithm 4.6 for an arc $([u_i], [u_{i+1}]) \in [A]$. Then*

$$\begin{aligned}\bar{t}^{[u_i]} \leq 1 + \bar{t}(V^{act} \cap ([u_{i+1}] \times T)) + \bar{d}([u_{i+1}], [u_{i+2}]) - \underline{d}([u_{i+1}], [u_{i+2}]) \\ - 3\underline{d}([u_i], [u_{i+1}]) + 2\bar{d}^{[u_{i+1}]}_{[u_i]} - \underline{\delta}.\end{aligned}$$

where $\underline{\delta} = \min\{\underline{\delta}_{[u_i]} \colon i = 1, \ldots, k\}$ with $\underline{\delta}_{[u]}$ defined in Proposition 4.45.

Proof. From Observation 4.48 and A4.6.5 we obtain $T^{[u_i]} = T^{[u_i][u_{i+1}]} = \{\hat{t}^{[u_i][u_{i+1}]}\}$. The value of this time shift can be bounded using A4.6.1–A4.6.4,

$$\begin{aligned}\underline{\tau}_{[u_i][u_{i+1}]} \leq \hat{t}^{[u_i][u_{i+1}]} = \max T^{[u_i][u_{i+1}]} &\leq t_{[u_i][u_{i+1}]} - \underline{d}([u_i], [u_{i+1}]) \\ &= \underline{\tau}_{[u_i][u_{i+1}]} + \bar{d}^{[u_{i+1}]}_{[u_i]} - \underline{d}([u_i], [u_{i+1}]).\end{aligned} \tag{4.18}$$

By A4.6.6 and $[G]$ being a path we have

$$\Delta_{[u_i][u_{i+1}]} \leq \bar{d}([u_{i+1}], [u_{i+2}]) - \underline{d}([u_{i+1}], [u_{i+2}]) - \underline{d}([u_i], [u_{i+1}]).$$

In order to bound the value $\bar{t}^{\hat{t}^{[u_i][u_{i+1}]}}_{[u_i][u_{i+1}]}$ defined in (4.15) let $(u', t_{u'}) \in V^{[u_i]}_{\hat{t}^{[u_i][u_{i+1}]}}$ and $(u', t_{u'})$ $(v', t_{v'}) \in A^T$ with $v'v' \in A$, and recall that for $(u', t_{u'}) \in V^{[u_i]}_{\hat{t}^{[u_i][u_{i+1}]}}$ there is a $(u', \delta_{u'}) \in V^{[u_i]}$ with $t_{u'} = \hat{t}^{[u_i][u_{i+1}]} + \delta_{u'}$. Then

$$\begin{aligned}
t_{v'} &\geq t_{u'} + \underline{d}([u_i], v') \\
&\geq t_{u'} + \min_{\substack{v'' \in [u_{i+1}], \\ v''v'' \in A}} \underline{d}([u_i], v'') \\
&\geq \hat{t}^{[u_i][u_{i+1}]} + \underline{\delta}_{[u_i]} + \min_{\substack{v'' \in [u_{i+1}], \\ v''v'' \in A}} \underline{d}([u_i], v'') \\
&\overset{(4.18)}{\geq} \underline{\tau}_{[u_i][u_{i+1}]} + \underline{\delta}_{[u_i]} + \min_{\substack{v'' \in [u_{i+1}], \\ v''v'' \in A}} \underline{d}([u_i], v'') \\
&\overset{A4.6.1}{\geq} \bar{t}(V^{\text{act}} \cap ([u_{i+1}] \times T)).
\end{aligned}$$

This proves that for each choice of $v' \in [u_{i+1}]$ with $v'v' \in A$ and $(u', t'_u) \in V^{[u_i]}_{t_{[u_i]}}$ the inner minimum in (4.15) will be attained for some arc $(u', t_{u'})(v', t_{v'}) \in A^T$ with $t_{v'} - t_{u'} = \underline{d}(u', v')$. Therefore, by (4.15),

$$\begin{aligned}
\bar{t}^{\hat{t}^{[u_i][u_{i+1}]}}_{[u_i][u_{i+1}]} &= \max_{\substack{v' \in [u_{i+1}], \\ v'v' \in A}} \max_{(u', t_{u'}) \in V^{[u_i]}_{\hat{t}^{[u_i][u_{i+1}]}}} (t_{u'} + \underline{d}(u', v')) \\
&= \hat{t}^{[u_i][u_{i+1}]} + \max_{\substack{v' \in [u_{i+1}], \\ v'v' \in A}} \max_{(u', \delta_{u'}) \in V^{[u_i]}} (\delta_{u'} + \underline{d}(u', v')) \qquad (4.19) \\
&\overset{(4.18)}{\leq} \underline{\tau}_{[u_i][u_{i+1}]} + \bar{d}^{[u_{i+1}]}_{[u_i]} - \underline{d}([u_i], [u_{i+1}]) + \bar{\delta}_{[u_i]} + \max_{\substack{v' \in [u_{i+1}], \\ v'v' \in A}} \max_{(u', \delta_{u'}) \in V^{[u_i]}} \underline{d}(u', v') \\
&\overset{A4.6.2}{\leq} \underline{\tau}_{[u_i][u_{i+1}]} + \bar{d}^{[u_{i+1}]}_{[u_i]} - \underline{d}([u_i], [u_{i+1}]) + \bar{d}^{[u_{i+1}]}_{[u_i]} \\
&\overset{(4.17)}{\leq} 1 + \bar{t}(V^{\text{act}} \cap ([u_{i+1}] \times T)) - \underline{\delta} - 2\underline{d}([u_i], [u_{i+1}]) + 2\bar{d}^{[u_{i+1}]}_{[u_i]}.
\end{aligned}$$

Now observe that $\bar{t}^{\hat{t}^{[u_i][u_{i+1}]}}_{[u_i][u_{i+1}]} + \Delta_{[u_i][u_{i+1}]} \leq \bar{\tau}$ where

$$\begin{aligned}
\bar{\tau} = 1 + \bar{t}(V^{\text{act}} \cap ([u_{i+1}] \times T)) + \bar{d}([u_{i+1}], [u_{i+2}]) - \underline{d}([u_{i+1}], [u_{i+2}]) \\
- 3\underline{d}([u_i], [u_{i+1}]) + 2\bar{d}^{[u_{i+1}]}_{[u_i]} - \underline{\delta}.
\end{aligned}$$

Furthermore, using $\bar{\delta}_{[u_i]} \overset{A4.6.2}{\leq} \bar{d}^{[u_{i+1}]}_{[u_i]} - \underline{d}([u_i], [u_{i+1}])$, we get

$$\begin{aligned}
\bar{t}(V^{[u_i]}_{\hat{t}^{[u_i][u_{i+1}]}}) &= \hat{t}^{[u_i][u_{i+1}]} + \bar{\delta}_{[u_i]} \\
&\overset{(4.18)}{\leq} \underline{\tau}_{[u_i][u_{i+1}]} + \bar{d}^{[u_{i+1}]}_{[u_i]} - \underline{d}([u_i], [u_{i+1}]) + \bar{\delta}_{[u_i]} \\
&\overset{(4.17)}{\leq} 1 + \bar{t}(V^{\text{act}} \cap ([u_{i+1}] \times T)) - \underline{\delta} - 3\underline{d}([u_i], [u_{i+1}]) + 2\bar{d}^{[u_{i+1}]}_{[u_i]} \leq \bar{\tau}.
\end{aligned}$$

Therefore A4.6.7 and A4.6.8 yield $\bar{\tau}^{[u_i][u_{i+1}]} \leq \bar{\tau}$, $\bar{t}^{[u_i][u_{i+1}]} \leq \bar{\tau}$ and, due to $T^{[u_i]} = T^{[u_i][u_{i+1}]} = \{\hat{t}^{[u_i][u_{i+1}]}\}$, also $\bar{t}(V^{[u_i]}_{\max T^{[u_i]}}) \leq \bar{\tau}$. This proves that all terms in A4.3.5 satisfy the bound as claimed. □

Thus, if waiting nodes follow in reasonably close succession and these waiting nodes do indeed require more time to proceed to the next clone node than the other nodes of the clone, then the generated subnetwork will exceed G^{act} only within a reasonably small time offset.

4.4.5 Summary

Dynamic graph generation offers a technique for the exact solution of shortest path problems on time expanded acyclic networks with infinite or large time horizons, where the basic cost structure ensures the existence of finite paths also in the presence of additional dual prices (conditions (C1)–(C3)). Typically, such dual prices arise in connection with coupling constraints in Lagrangian relaxation or column generation.

Dynamic graph generation relies on solving a sequence of shortest path problems on *valid subnetworks* (Definition 4.5, Algorithm 4.1) with finite time horizon, that allow to recognise whether the current network excludes shortest paths that exceed the current time horizon and provide the next extension at the same time. We analysed this approach under the assumption of an infinite time horizon and showed its finiteness under reasonable assumptions (Lemma 4.15,Lemma 4.16).

For the case that the cost function is known to prefer early paths to late paths (condition (C)), a valid subnetwork is obtained by including, on top of all paths computed in previous steps (their closure is G^{act}), all G^{act}-irreducible paths (Definition 4.27, Corollary 4.28). We described two algorithmic approaches for constructing subnetworks that contain all G^{act}-irreducible paths.

The first includes, on top of G^{act}, a corridor of unused "fastest" paths (these depart from the source as late as possible in order to reach the sink at a given time level without intermediate waiting) that must be intersected by any path that exceeds the current time horizon (Theorem 4.30).

The second improves on this in two ways. First, instead of forming a corridor along each possible node sequence, it constructs a corridor along clone routes via precomputed interception sets (Definition 4.31) and sets of admissible time shifts (Definition 4.37). This may allow to replace paths offering only slow connections by equivalent faster paths, thereby reducing the time span needed for guaranteeing that the corridor lies outside G^{act}. Second, in the presence of waiting nodes within clone nodes, the required number of time steps between corridor and G^{act} may be reduced further by shifting the interception sets to earlier time steps than required for a fastest continuation towards the sink. This may entail that some irreducible paths meet the corridor only on exiting but not on reentering G^{act} which leads to irreducible nodes (Definition 4.41). Still, a valid subnetwork based on the improved corridor and all irreducible nodes can be computed in time linear in the number of clone arcs $|[A]|$ times the precomputed maximum cardinality of the admissible time shift sets, see Theorem 4.49. If the algorithm is applied to a graph that

is a clone path, the corridor provably stays close to G^{act} whenever waiting is allowed (Observation 4.50,Observation 4.51).

4.5 Dynamic Graph Generation in the TTP

In this section we investigate how dynamic graph generation can be applied in the solution process for our models for the TTP, (TTP-clq) and (TTP-cfg). As discussed in Chapter 3, our solution process in based on Lagrangian relaxation of the coupling constraints (column generation would also work), thus in each iteration we have to solve a shortest path problem in each network. In order to apply the dynamic graph generation approach developed in this chapter, we have to ensure that the train graphs have the desired structure and the objective functions satisfy (C1) as well as either (C2) and (C3) or (C2'). Furthermore if we want to use the shifted corridor, the objective function must satisfy (C).

First we consider the departure constraints (2.5). As mentioned in Remark 2.5, they can be modelled by setting the correspond arc weights to $+\infty$. In practice, we simply leave them out of the stored current subgraph G^{cur} completely. This corresponds to setting $h^{\text{aug}}(a) = +\infty$ for all forbidden arcs a. Of course, then one has to ensure that G^{cur} contains at least one path, because otherwise the subproblem ($DyGG^{\text{cur}}(G^{\text{cur}},h^{\text{aug}})$) would be infeasible. This can be implemented by simply dealing with G^{act} as if it contains all forbidden arcs, because then the generated corridor (and thus G^{cur}) will indeed contain sufficient further nodes and arcs so that this path exists. In particular, the departure constraints do not appear explicitly as coupling equality constraints.

Next we investigate (TTP-clq). In this model all time expanded networks are train graphs, the clone partition being defined by the underlying route graph (see Remark 4.3), so fix some $r \in R$. In other words, choosing the clone nodes

$$[(u, b_u)] := \{(u, b') \in \{u\} \times B \colon (u, b') \in V_r\}$$

for any node $(u, b_u) \in G_r$ implies a valid clone partition satisfying (K1) and (K2). Condition (K3) can easily be satisfied, possibly by adding an artificial source resp. sink node (in the following we will assume for simplicity, that the base train graphs do already include a source and sink node). All coupling constraints are inequalities with non-negative coefficients implying (C1) (see Remark 4.11). By definition of the weights (2.9) resp. (2.10) the objective value of a path $P = (\hat{u}, t_{\hat{u}}) \dots (u, t_u)(\check{u}, t_{\check{u}})$ is at least

$$h(P) \geq h((u, t_u)(\check{u}, t_{\check{u}})) = \omega(t_{\check{u}})$$

fulfilling (C2'). This allows to use the basic dynamic graph generation approach without shifting. Furthermore the next result shows that the objective functions satisfy (C), so we can use the shifting algorithm for all train graphs.

Lemma 4.52 *The objective functions defined by the weights (2.9) and (2.10) fulfil (C).*

Proof. Fix some train $r \in R$ and let $P^p \in \mathcal{P}$, $p = 1, 2$, be two paths with $P^1 \leq_T P^2$. Each node in V_r^T has the form $((u, b_u), t_u) \in (\bar{V}_r \times B) \times T$ and $[(u, b_u)] = [(v, b_v)] \iff u = v$. $P^1 \leq_T P^2$ implies by definition that $[P^1] = [P^2]$, thus both paths have the form

$$P^p := ((u_i, b_i^k), \underline{t}_i^k) \dots ((u_i, b_i^k), \bar{t}_i^k)((u_{i+1}, b_{i+1}^k), \underline{t}_{i+1}^k) \dots ((u_j, b_j^k), \bar{t}_j^k),$$

for $p = 1, 2$. Again by $P^1 \leq_T P^2$ it holds for all $k \in \{i, \dots, j\}$,

$$\underline{t}_k^1 \leq \underline{t}_k^2. \tag{4.20}$$

Now using the definition of the weights (2.9) and (2.10), the objective values of the paths depend only on arrival times $\underline{t}_k^p$ of the transfer arcs, *i. e.* for $p = 1, 2$ it is

$$h(P^p) = \sum_{k=i}^{j-1} w_{((u_k, b_k^p), \bar{t}_k^p)((u_{k+1}, b_{k+1}^p), \underline{t}_{k+1}^p)},$$

because all other weights are zero. Furthermore the definition of the penalty functions together with (4.20) imply

$$\forall\, k \in \{i, \dots, j-1\} \colon w_{((u_k, b_k^1), \bar{t}_k^1)((u_{k+1}, b_{k+1}^1), \underline{t}_{k+1}^1)} \leq w_{((u_k, b_k^2), \bar{t}_k^2)((u_{k+1}, b_{k+1}^2), \underline{t}_{k+1}^2)},$$

proving $h(P^1) \leq h(P^2)$. □

Now consider model (TTP-cfg). In these models we have a second kind of time expanded network, the configuration networks $G_a^T = (V_a^T, A_a^T)$, $a \in A^I$. Furthermore, in addition to the capacity constraints, which are inequalities, the model contains the configuration constraints (2.8), which are *equalities*. Whereas the capacity constraints do not harm the requirement (C1) of the active objective function h^{aug} (Remark 4.11), the Lagrange multiplier of a configuration constraint may take any real value, thus possibly causing the augmented cost value of some arc to be less than its base value, *i. e.* $h^{\text{aug}}(a)_r < h_r(a)$.

First we consider train graphs and show that (C1) still holds if we apply a standard first order method. Fix again some $r \in R$ and choose an arbitrary transfer arc $a \in A_r^T$ and its corresponding configuration arc $a' = \text{cfg}(a)$. We show that the Lagrange multiplier associated with the configuration constraint

$$x_a = x_{a'}$$

can only be non-zero if a is contained in the active subgraph G^{act}. This would suffice to satisfy (C1), because only a non-zero Lagrange multiplier can induce negative augmented costs. We assume that the initial value of the Lagrange multiplier is zero (it is standard for first order methods to start with $y = 0$ as initial point when applied in Lagrangian relaxation). Now suppose $a \notin G^{\text{act}}$ in an arbitrary iteration of the algorithm. This means that a has never be contained in an optimal path computed in an earlier iteration because otherwise Algorithm 4.1 would have added a to G^{act}. Consequently the variable x_a has had the value zero in all earlier iterations. If we can show that the corresponding configuration variable $x_{a'}$ has also had been zero in all earlier iterations, we are done (see below). In

this case the constraint $x_a = x_{a'}$ would have been satisfied and the corresponding entry in the subgradient would have always been zero. But a first order method (as described in Section 3.5) always looks for the next candidate (the next Lagrange multiplier vector) in the span$\{g_1, g_2, \dots\}$ of the subgradients returned so far . Because all those subgradients are zero in the component corresponding to the configuration constraint of a, the next candidate is also zero in that component. Consequently, as long as a is not contained in an optimal path, the Lagrange multiplier of its configuration constraint will remain zero and never induce non-zero augmented costs. Hence $h^{\text{aug}}(a) \geq h(a)$ holds as long as $a \in G^{\text{act}}$.

It remains to show that $x_{a'} = 0$ as long as $x_a = 0$. This does not necessarily hold but depends on the shortest path algorithm used to solve the subproblem in the configuration network. Obviously, as long as the Lagrange multiplier remains zero, no augmented costs are induced for the arc a' in the configuration network. The basic objective function in configuration networks is $h = 0$, hence the augmented cost value of a' is also zero. Now suppose a' is contained in an optimal path of the configuration network. This path corresponds to a configuration. Because the headway times fulfil a triangle inequality (see Section 2.2) we can simply drop a' without affecting validity of the configuration, and because $h^{\text{aug}}(a') = 0$, without changing the value of the optimal solution. This means, if the optimal solution of the shortest path subproblem always returns a path so that all configurations arcs on this path have *negative* costs (arcs with positive costs can be skipped as well), we have indeed that $x_{a'} = 0$ as long as $x_a = 0$.

Finally we have to consider configuration networks. These networks do not fit in our framework, because they are not acyclic. Fortunately they have a very simple base cost function $h = 0$ and a nice structure so that the construction of a "valid subnetwork" is quite easy in this case. The discussion above shows that only arcs with non-zero (even negative) costs have to be considered and these are only configuration arcs corresponding to train arcs that have been used in an earlier iteration. So we only have to include exactly those configuration arcs and their connecting arcs.

5 Asynchronous Parallel Bundle Algorithm

5.1 Introduction

One critical part of the solution of combinatorial optimisation problems is the computation of bounds on the optimal value. These bounds are typically computed by solving a linear or convex relaxation of an integer programming formulation. Linear relaxations can usually be solved quite efficiently using standard solvers for small to medium sized problems. But large scale problems often require different solution approaches. One of them is Lagrangian relaxation, which leads to a convex, non-smooth optimisation problem. This optimisation problem is then often solved using a first order method like a subgradient or bundle method. Lagrangian relaxation is also the solution approach that we employed for our practical application, the Train Timetabling Problem (see Chapter 2). In Chapter 3 we outlined the Lagrangian relaxation approach as it is used for the TTP.

In this chapter we focus on an optimisation algorithm for solving Lagrangian relaxations of integer programming problems. Taking a classical bundle method as a basis, the goal is to develop an advanced asynchronous and parallel bundle method exploiting the structural dependencies in a Lagrangian relaxation formulation. For the sake of a comprehensive presentation and in order to emphasise the general applicability of this approach beyond the TTP, we repeat the setting once again (see Section 3.2).

We consider a general combinatorial optimisation problem in the following form

Problem Let R be a finite set and for each $r \in R$

- $\mathcal{X}_r \subseteq \mathbb{R}^{n_r}$, $n_r \in \mathbb{N}$ a compact *ground set*,
- $h_r \colon \mathcal{X}_r \to \mathbb{R}$ *primal objective functions*,

and set $n = \sum_{r \in R} n_r$. Furthermore let $M = \{1, \dots, m\}$ be a (finite) set and

- $C \in \mathbb{R}^{M,n}$,
- $b \in \mathbb{R}^M$.

Then the *primal optimisation problem* reads

$$\text{(P)} \qquad \begin{aligned} &\text{maximise} && \sum_{r \in R} h_r(x_r) \\ &\text{subject to} && x_r \in \mathcal{X}_r, && r \in R, \\ & && \sum_{r \in R} C_{\bullet,r} x_r = b. \end{aligned}$$

Without loss of generality, we assume that each row of C contains at least one non-zero coefficient, *i. e.*

$$\forall\, j \in M \colon C_{j,\bullet} \neq 0. \tag{5.1}$$

Remark 5.1 For simplicity, we consider only equality constraints. The algorithm to be developed in this chapter can be applied to inequality constraints as well, but this would only complicate the presentation and would not provide any new insights. We will refer to inequality constraints at appropriate points in the presentation.

Note that our implementation does support inequality constraints, which is required for our TTP models.

Problems of the kind (P) are not specific to TTP models but appear generically in connection with multi-commodity flow models in scheduling and planning applications. Indeed, Lagrangian relaxation and decomposition has proved to be a valuable tool, see, *e. g.*, [6, 37, 64, 65] for an overview on Lagrangian relaxation and applications. One example of typical applications is the planning of operation of power plants, see, *e. g.*, Zhang et al. [95] or Mezger and Almeida [74]. Sagastizábal and Solodov [85] used bundle methods for the planning of future constructions of new power plants in conjunction with stochastic programming. Indeed, bundle methods have been applied successfully to several multi-stage stochastic programming problems, see Oliveira et al. [79] and Schultz [88]. Problems concerning large networks as in telecommunication (Lemaréchal et al. [67], Bley et al. [5]), network design (Crainic et al. [23]) or more generic multi-commodity flows (Babonneau et al. [3]) are also canonical candidates for Lagrangian relaxation and bundle methods. Closer to our application, the TTP, are scheduling problems (Gu [46]), vehicle routing problems (Borndörfer et al. [8]) and inventory management problems (Helmberg and Röhl [53]), which employ Lagrangian relaxation of coupling constraints between network subproblems.

The increasing problem sizes in these areas as well as the availability of cheap parallel computing hardware require the development of decomposition techniques not only on

the modelling but also on the algorithmic side. We contribute to this by proposing an asynchronous parallel approach for Lagrangian relaxation under the rather natural assumption in this setting, that most of the coupling constraints either couple only few subproblems or, if they couple many, coupling actually *affects* just a few subproblems.

Formally, we form the *Lagrangian* $L(x, y)$ by

$$L\colon \mathcal{X} \times \mathbb{R}^M \to \mathbb{R},$$
$$L(x, y) = \sum_{r \in R} h_r(x_r) + \langle y, b - Cx \rangle$$

where $\mathcal{X} = \bigtimes_{r \in R} \mathcal{X}_r$. Then the Lagrangian relaxation of (P) is

Problem

(LD)
$$\begin{aligned} \text{minimise} \quad & f(y) \\ \text{subject to} \quad & y \in \mathbb{R}^M \end{aligned}$$

where
$$f(y) := \max\{L(x, y)\colon x \in \mathcal{X}\}.$$

The function f is convex and typically non-smooth. A subgradient or bundle method requires the evaluation of f at certain points $y \in \mathbb{R}^M$, which should provide the function value $f(y)$ as well as a subgradient $g \in \partial f(y)$. By definition we have

$$f(y) = \langle b, y \rangle + \sum_{r \in R} \max\{h_r(x_r) - \langle C^T_{\bullet,r} y, x_r \rangle\colon x_r \in \mathcal{X}_r\}.$$

Given optimal solutions $x^*_r \in \mathcal{X}_r$, $r \in R$, of the subproblems

(Sub_r)
$$\begin{aligned} \text{maximise} \quad & h_r(x_r) - \langle C^T_{\bullet,r} y, x_r \rangle \\ \text{subject to} \quad & x_r \in \mathcal{X}_r, \end{aligned}$$

for a certain $y \in \mathbb{R}^M$, the function value and a subgradient can easily be provided via the relations

$$f(y) = \langle b, y \rangle + \sum_{r \in R} \left(h_r(x^*_r) - \langle C^T_{\bullet,r} y, x^*_r \rangle \right),$$
$$g(x^*_r) = b - \sum_{r \in R} C_{\bullet,r} x^*_r \in \partial f(y).$$

Throughout this chapter we assume that the subproblems (Sub_r) can be solved efficiently for any $y \in \mathbb{R}^M$. The optimal solution of (LD) is, in general, an upper bound on the

optimal solution of (P). In fact, its optimal value is equal to the optimal value of the "convexified" relaxation of (P) (cf. Section 3.3)

$$\text{(conv P)} \qquad \begin{aligned} &\text{maximise} && \sum_{r\in R}(\operatorname{conc} h_r)(x_r) \\ &\text{subject to} && x_r \in \operatorname{conv} \mathcal{X}_r, \qquad r \in R, \\ & && \sum_{r\in R} C_{\bullet,r} x_r = b, \end{aligned}$$

where $\operatorname{conc} h_r$ denotes the "concave hull" of h_r, $r \in R$.

The computationally most expensive part of a first order method solving (LD) is usually the solution of the subproblems. In each iteration, each subproblem must be solved w. r. t. a certain point $y \in \mathbb{R}^M$. Fortunately, the subproblems (Sub_r) are independent from each other. Once the point of evaluation $y \in \mathbb{R}^M$ has been determined, each subproblem can be solved in isolation producing an optimal solution $x_r \in \mathcal{X}_r$. This means that the evaluation of the subproblems can be performed *in parallel*, *i. e.* simultaneously on several parallel processors so that each processor solves one subproblem (here processor may refer to a CPU core, a single CPU or a node in a parallel computer depending on the computing environment). The purpose of this chapter is to extend this general and simple parallelisation approach even further.

In real applications, the subproblems do not have all the same difficulty. Often there are several easy subproblems, some more challenging subproblems and few really hard subproblems. The hardness may arise from the structure of the subproblem itself or from their dependency on the coupling constraints. This means, some subproblems are heavily influenced by the coupling constraints while others are not. Because of these differences, some of the subproblems require significantly more effort to be solved or need to be solved more often than others during the course of the first order method. But unfortunately, standard first order methods do not take these differences into account: in each iteration all subproblems must be solved exactly once.

In this chapter, starting from a standard, classical proximal bundle method, we develop a fully *parallel* and *asynchronous* bundle method, that addresses these issues. Instead of solving all subproblems in each iteration, the algorithm dynamically selects a subspace of the variables $y \in \mathbb{R}^M$ together with the subproblems being influenced by these variables. Then an independent subprocess optimises f on the selected subspace by considering only the associated subproblems. The main algorithm does not select only one subspace but several independent subspaces in parallel. In addition, the main algorithm does not wait until all subproblems are solved to complete an iteration. Instead, if some subspace problem has been solved, the main algorithm updates the global data immediately with the subspace solution. Thus, the main algorithm behaves in an asynchronous manner, not waiting for subproblems that require more computational effort than others. Optimising on subspaces asynchronously without some synchronising mechanism may easily endanger convergence of the algorithm. The progress made on some subspace may worsen the situation on another subspace. Therefore our algorithm detects automatically such bad interdependencies between subspaces. Future subspace selections take care of this

information so that variables that possess such dependencies are always included together in the same subspace.

The increasing importance of parallel computing hardware has got significant interest in the literature during the last decade. A central concept for many parallelisation approaches is decomposition of a large scale optimisation problem into smaller, independent subproblems, see Boyd et al. [10] for an overview of decomposition techniques. In particular Lagrangian relaxation (*e. g.*, Kiwiel [58],) and Dantzig-Wolfe decomposition (Dantzig and Wolfe [24]) have successfully been applied in many fields, a comparison of these two decomposition techniques can be found in Briant et al. [12].

Once a problem is decomposed into independent subproblems, a standard approach to parallelisation is to distribute the evaluation of these subproblems among several processors. In order to highlight the difference of our asynchronous scheme to existing approaches we shortly review parallel algorithms in the literature. The already mentioned standard approach distributes the subproblems in a queen-worker manner among several worker processes, the results of the subproblems are combined by a queen process to a new point in an iterative scheme. A computational study of this approach for a bundle algorithm can be found in Medhi [72], a parallel subgradient algorithm is discussed in Hanafi et al. [47] and Tebboth [92] investigates Dantzig-Wolfe decomposition.

A first class of algorithms are parallel coordinate descent methods, which optimise along coordinate directions instead of general (subgradient-)directions. In Richtárik and Takáč [82] this approach has been developed into a randomised parallel algorithm, that optimises in several coordinate directions independently in parallel. This algorithm makes some structural assumptions on the objective function like smoothness and partial separability (*i. e.*, the objective function can be written as a sum of functions, each of which depends only on a subset of variables). The problems that we consider are in general non-smooth and partial separability is only required for the basic algorithm described in Section 5.3, but not for the extended algorithm in Section 5.4. Furthermore our algorithm does not necessarily optimise along coordinate directions only. Instead the algorithm selects dynamically a whole subspace generated by several coordinate directions and determines the next iterate by considering a restriction of the original problem to this subspace.

Another class of algorithms are variable transformation algorithms for unconstrained (Fukushima [38]) and constrained (Sagastizábal and Solodov [84]) optimisation problems. In each iteration they select a set of subspaces either along coordinate directions [84] or more general directions [38] such that the subspaces span the whole space and then compute a new candidate point on each subspace where each subspace problem can be solved in an independent process. Afterwards a new global iterate is derived from the candidate points. Usually these approaches require smoothness of the objective function but can also be applied to non-smooth optimisation problems using smoothing techniques as, *e. g.*, the Moreau-Yosida regularisation (Meng et al. [73]).

A successful approach to non-smooth optimisation uses incremental subgradient methods. These methods change y in each major iteration incrementally through a sequence of ω steps. In each step y is modified according to a subgradient direction of a single subproblem. Nedić et al. [77] extend this approach to a fully parallel and asynchronous approach where the sequences in which the subproblem computations are started and

in which y is modified may differ. In particular, when the candidate computed by a subproblem is considered as a new point, the current point y may differ from the point when that subproblem was started (in contrast to the non-parallel case). Nedić et al. [77] prove the convergence of this approach under the assumption that the number of updates of y between the start of the subproblem's processing and the consideration of the subproblem's result are bounded by a constant. This implies that the overall number of evaluations is asymptotically equal for each subproblem, which is also the case for the synchronised parallel approaches above.

Other parallel approaches exploit the explicit static structure of convex optimisation problems. Those so called splitting methods solve easier subproblems generated by the corresponding augmented Lagrangian function, which may be solved alternatingly or in parallel, and combine the results to solutions of the original problem in some iterative scheme, see, *e. g.*, Goldfarb and Ma [42, 43], He et al. [49], Nayakkankuppam [75], Necoara and Suykens [76].

The subspace selection of our approach is somewhat similar to the variable distribution approaches [32, 38, 84], but in contrast we do not require to select a set of subspaces that span the whole space. Furthermore there is no global synchronisation step as in [42, 43] and no requirement on how often a certain subproblem must be evaluated. Indeed, some subproblems may require considerably fewer evaluations than others and this is a main source of efficiency. In contrast to the algorithms proposed in the papers above, there is no regularisation condition or synchronisation to ensure global convergence. Instead, convergence is guaranteed only by dependency analysis between the subspace evaluations.

An algorithm, in which the subproblems may be evaluated in a largely asynchronous way, is based on ball-step methods (which are also subgradient methods) in Kiwiel and Lindberg [59]. There the only requirement is that each subproblem is evaluated "sufficiently often", although the relative number of evaluations between different subproblems is largely unrestricted.

In contrast our algorithm is based on bundle methods, see Bonnans et al. [6] for an introduction and Hiriart-Urruty and Lemaréchal [55], Lemaréchal et al. [66] for a detailed exposition on bundle methods. We give a short introduction to bundle methods in Section 5.2.

Besides the parallelisation approaches described above there is another natural possibility to exploit modern parallel hardware, which is of completely different type. In each iteration the bundle method has to solve a convex quadratic subproblem, precisely (5.3) below. These kind of subproblems are typically solved using numerical methods like interior point methods. Such methods rely on common linear algebra subroutines like matrix-vector multiplication, which are well suited for efficient implementations on parallel hardware, see, *e. g.*, Gondzio and Grothey [44]. In this chapter we do not deal explicitly with the solution of the quadratic subproblem. Instead we rely on standard software [56] to solve the quadratic subproblem, *i. e.*, we consider the quadratic subproblem to be solved by a black box solver. In fact, the quadratic subproblems could be solved by any parallel solver as well. However, in our application, the major computational effort was the evaluation of the subproblems, so an increased efficiency of the quadratic subproblem won't yield much better performance of the overall algorithm.

This chapter is divided into three main parts. The first part Section 5.2 shortly introduces a classical proximal bundle method, which is the starting point for our development. In the second part in Section 5.3 we develop the basic asynchronous parallel bundle method. Here we discuss the major ideas and the algorithm and give some convergence results. The presentation of the algorithm is relatively simple, but it may not yet be suitable for practical applications. This is because that algorithm assumes that each single constraint couples only few subproblems, whereas in practice many constraints typically couple a lot of subproblems. However, the number of subproblems that are *actually* influenced during the algorithm by a certain constraint is often rather small (this means, although some constraint may have non-zero coefficients in a lot of subproblems, most of the subproblems are not really influenced by this constraint). In the third part in Section 5.4 we extend the basic algorithm to take these interactions into account. Similarly to the dependency detection between subspaces, the algorithm automatically discovers whether a certain constraint influences some subproblem and respects this relation in the future. This additional step increases the notational burden a lot, but leads to an algorithm that is better suited for practical applications. As before, we present the algorithm together with some convergence results.

The results presented in this chapter are taken from [33], partially verbatim. The presentation in this thesis differs in following aspects from the paper. First, we added a discussion of standard bundle methods in Section 5.2. Each main step of the algorithm is now discussed in its own subsection, which contains more explanations. The presentation of consistency of the global data in the extended parallel bundle method (Section 5.4) differs slightly from the paper version in order to improve readability. Furthermore we added a section that relates the parallel bundle method to the motivating TTP. The notation has been adapted so as to match the notation used in this thesis.

5.2 Bundle Methods

In this section we give a short introduction to (proximal) bundle methods, which form the basis of our parallel bundle method. Bundle methods are described in great detail in [6, 55, 66]. The notation used in this section is intentionally similar to the notation used in the previous chapters in order to emphasise the relation to our TTP models. We will also reuse the notation later in this chapter when we describe the parallel bundle method. Although some terms may not be exactly equivalent, we hope that this decision helps to improve the understanding of the relations between the different chapters of this work.

We consider an unrestricted optimisation problem

$$\text{(Cvx)} \qquad \text{minimise}\{f(y)\colon y \in \mathbb{R}^M\}.$$

The bundle method we consider is a first order method, so we assume that the function is given by a first order oracle that, for a specified point $y \in \mathbb{R}^M$, returns

(i) the function value $f(y)$ and

(ii) a subgradient $g \in \partial f(y)$.

The classical proximal bundle algorithm then works as follows. Given a current iterate $\hat{y}^k$, the so called *centre of stability*, the bundle method manages a *model* $\hat{f}^k$ of the function

$$\forall\, y \in \mathbb{R}^M \colon \hat{f}^k(y) \leq f(y).$$

The model can be rather general, but for our purposes we always assume it being a cutting plane model, *i. e.*, given a finite set (a compact set would also be sufficient)

$$\widehat{W}^k = \{(l^i, g^i) \in \mathbb{R} \times \mathbb{R}^M \colon i \in I^k\}, \tag{5.2}$$

where I^k is a finite index set, the model is

$$\hat{f}^k(y) = \max\left\{l + \langle g, y\rangle \colon (l, g) \in \widehat{W}^k\right\}.$$

Note that $\hat{f}^k$ is convex because it is the maximum of convex functions. An important fact is that each subgradient $g \in \partial f(\bar{y})$ at a certain point $\bar{y} \in \mathbb{R}^M$ defines a linear model of f:

$$y \mapsto f(\bar{y}) + \langle g, y - \bar{y}\rangle,$$

or, equivalently,

$$y \mapsto l + \langle g, y\rangle \quad \text{where} \quad l = f(\bar{y}) - \langle g, \bar{y}\rangle.$$

That this affine function is really a global minorant of f follows from the convexity of f and the subgradient inequality

$$g \in \partial f(\bar{y}) \iff \forall\, y \in \mathbb{R}^M \colon f(y) \geq f(\bar{y}) + \langle g, y - \bar{y}\rangle.$$

The bundle method then computes the next *candidate* point $\bar{y}^k$ by determining the optimiser of the augmented model

$$\min_{y \in \mathbb{R}^M} \left\{\hat{f}^k(y) + \frac{u}{2}\|y - \hat{y}^k\|^2\right\}, \tag{5.3}$$

i. e.

$$\bar{y}^k = \operatorname*{argmin}_{y \in \mathbb{R}^M} \left\{\hat{f}^k(y) + \frac{u}{2}\|y - \hat{y}^k\|^2\right\}.$$

Here $u > 0$ is a fixed parameter, the *weight* of the augmenting term (in general this parameter may be changed in each iteration, but we will always assume it being fixed; see [55] for more details). Note that the candidate $\bar{y}^k$ is well defined because the augmented objective function

$$\hat{f}^k(y) + \frac{u}{2}\|y - \hat{y}^k\|^2 \tag{5.4}$$

is strictly convex and bounded from below. The function model together with the candidate predicts a certain decrease in the objective value if one moved the centre to $\bar{y}^k$, namely

$$\Delta^k = f(\hat{y}^k) - \hat{f}^k(\bar{y}^k).$$

However, the bundle method only accepts the candidate as a new centre if the *actual decrease* is sufficiently large compared to the predicted decrease

$$f(\hat{y}^k) - f(\bar{y}^k) \geq \varrho\Delta^k. \tag{5.5}$$

If the candidate satisfies (5.5), then the model is a "good" approximation of f in $\bar{y}^k$ and a *descent step* is performed, *i. e.* $\bar{y}^k$ is accepted as the next centre of stability $\hat{y}^{k+1} \leftarrow \bar{y}^k$. Otherwise the model is a "bad" approximation of f. In this case the centre remains unchanged, $\hat{y}^{k+1} \leftarrow \hat{y}^k$, but the model is improved in $\bar{y}^k$ by adding an appropriate linear minorant to the model. More precisely, let $\tilde{g}^k \in \partial f(\bar{y}^k)$ be a subgradient in $\bar{y}^k$ (returned by the first order oracle), then the linear minorant

$$y \mapsto \tilde{l}^k + \langle \tilde{g}^k, y \rangle$$

where

$$\tilde{l}^k = f(\bar{y}^k) - \langle \tilde{g}^k, \bar{y}^k \rangle,$$

is added to the model, *i. e.* $(\tilde{l}^k, \tilde{g}^k) \in \widehat{W}^{k+1}$. In its simplest form the new model is $\widehat{W}^{k+1} = \widehat{W}^k \cup \{(\tilde{l}^k, \tilde{g}^k)\}$, *i. e.*, the new minorant is added to the previous model. Because the model may become quite large (*i. e.* contains many minorants) this way, one usually compresses the model by removing some minorants or by aggregating them into a new minorant. In particular, because $\operatorname{conv} \widehat{W}^k$ is compact and (5.4) is strongly convex, by Lagrangian duality the augmented model is equivalent to (see, *e. g.*, [55, Theorem VII 4.3.1 and Chapter VII.4.4])

$$\max_{(l,g) \in \operatorname{conv} \widehat{W}^k} \min \left\{ l + \langle g, y \rangle + \frac{u}{2} \|y - \hat{y}^k\|^2 \colon y \in \mathbb{R}^M \right\}. \tag{5.6}$$

The inner minimisation problem can be solved explicitly for a given $(l, g) \in \operatorname{conv} \widehat{W}^k$ yielding

$$\bar{y}^k(l, g) = \hat{y}^k - \frac{1}{u} g.$$

It can be worked out (see, [6, Lemma 9.8]) that the unique optimiser $(\bar{l}^k, \bar{g}^k) \in \operatorname{conv} \widehat{W}^k$ defines the optimal solution of the original augmented model $\bar{y}^k = \bar{y}^k(\bar{l}^k, \bar{g}^k)$. Furthermore, this optimiser defines another minorant of f, the so called *aggregated minorant*

$$\bar{f}^k(y) = \bar{l}^k + \langle \bar{g}^k, y \rangle.$$

One important property is that the predicted decrease can be expressed in terms of the aggregated minorant. Because $(\bar{l}, \bar{g})$ optimises (5.6) with $\bar{y}^k$ being the optimiser of the inner minimisation it is

$$\hat{f}^k(\bar{y}^k) + \frac{u}{2} \|\bar{y}^k - \hat{y}^k\|^2 = \bar{f}^k(\bar{y}^k) + \frac{u}{2} \|\bar{y}^k - \hat{y}^k\|^2,$$

so it holds

$$\hat{f}^k(\bar{y}^k) = \bar{f}^k(\bar{y}^k) = \bar{l}^k + \langle \bar{g}^k, \bar{y}^k \rangle = \bar{l}^k + \langle \bar{g}^k, \hat{y}^k \rangle + \langle \bar{g}^k, \bar{y}^k - \hat{y}^k \rangle = \hat{f}^k(\hat{y}^k) - \frac{1}{u} \|\bar{g}^k\|^2,$$

hence

$$\Delta^k = f(\hat{y}^k) - \hat{f}(\bar{y}^k) = f(\hat{y}^k) - \bar{f}^k(\hat{y}^k) + \frac{1}{u}\|\bar{g}^k\|^2. \tag{5.7}$$

Furthermore, one can show [see 6, Chapter 9.3.2] that the aggregated minorant carries all necessary information of the previous model $\widehat{W}^k$ in the sense that choosing $\widehat{W}^{k+1} = \{(\tilde{l}^k, \tilde{g}^k), (\bar{l}^k, \bar{g}^k)\}$, *i. e.* consisting of only two minorants, the new minorant defined by the subgradient $\tilde{g}^k \in \partial f(\bar{y}^k)$ and the aggregated minorant, is sufficient to get a convergent algorithm.

The classical termination criterion is to stop when the predicted decrease is sufficiently small. In fact, if $\Delta^k < \varepsilon$ for some small $\varepsilon > 0$, then (5.7) implies that $\|\bar{g}^k\|$ is small, thus $\hat{y}^k$ can be seen as an approximate optimal solution. This termination criterion is rather weak, the point $\hat{y}^k$ can be arbitrarily far from an optimal solution, but seems to work well in practice. We will use this termination criterion as well.

Putting all together we get the classical bundle algorithm shown in Algorithm 5.1.

The following results establish the convergence of the described classical bundle method. We omit the proofs and refer to the literature, *e. g.* [6]. The first result discusses the case of an infinite number of descent steps, given that the algorithm does not terminate for termination precision $\varepsilon = 0$.

Theorem 5.2 *(see, e. g., Theorem 9.14 in [6])*
Suppose that Algorithm 5.1 with $\varepsilon = 0$ generates infinitely many descent steps and that $f^* := \lim_{k\in\mathbb{N}} f(\hat{y}^k) > -\infty$. Then $f^* = \min_{y\in\mathbb{R}^M} f(y)$. Furthermore, if $\operatorname{Argmin}_{y\in\mathbb{R}^M} f(y) \neq \emptyset$, then the sequence of stability centres $(\hat{y}^k)_{k\in\mathbb{N}}$ has at least one cluster point, which is optimal.

(Note that the proof in [6] shows, that *each* cluster point is optimal.)

The second result considers the case when Algorithm 5.1 generates only a finite number of descent steps.

Theorem 5.3 *(Theorem 9.15 in [6])*
Suppose that Algorithm 5.1 with $\varepsilon = 0$ generates only a finite number of descent steps with $\hat{y}^{k_0}$, $k_0 \in \mathbb{N}$ being the last centre, i. e. $\hat{y}^k = \hat{y}^{k_0}$ for all $k \geq k_0$. Then the following holds:

- *The sequence $(\Delta_k)_{k>k_0}$ is decreasing and tends to 0.*
- *The last centre $\hat{y}^{k_0}$ minimises f.*

In this chapter we will consider the case that f is the Lagrangian dual function, *i. e.* problem (Cvx) is of the form (LD). In this case each point $x \in \mathcal{X}$ defines a minorant

$$y \mapsto h(x) + \langle g(x), y\rangle$$

Algorithm 5.1: ClassicalBundle

Input :

- function f given by a first order oracle
- termination precision $\varepsilon > 0$,
- descent parameter $\varrho \in (0,1)$,
- weight $u > 0$.

Output: approximate optimal solution $y^* \in R^M$

`// Initialisation`

Set $k \leftarrow 0$, $\hat{y}^0 \leftarrow 0$

Compute $(f^0, \tilde{g}^0) \leftarrow f(\hat{y}^0)$

Set initial model $\hat{f}^0(y) = f^0 + \langle \tilde{g}^0, y - \hat{y}^0 \rangle$

Choose a descent parameter $\varrho \in (0,1)$

`// Computation`

loop

`// Compute aggregated minorant by`

$$(\bar{l}^k, \bar{g}^k) = \operatorname*{argmax}_{(l,g) \in \operatorname{conv} \widehat{W}^k} \min_{y \in \mathbb{R}^M} \left\{ l + \langle g, y \rangle + \frac{u}{2} \|y - \hat{y}^k\|^2 \right\}$$

`// Compute next candidate by`

$$\bar{y}^k = \hat{y}^k - \frac{1}{u} \bar{g}^k$$

`// Termination test`

Evaluate $(\bar{f}^k, \tilde{g}^k) \leftarrow f(\bar{y}^k)$

Set $\Delta^k \leftarrow f^k - \hat{f}^k(\bar{y}^k)$

if $\Delta^k < \varepsilon$ **then**

set $y^* \leftarrow \hat{y}^k$

return y^*

`// Descent test`

if $\Delta^k < \varrho \cdot (f^k - \bar{f}^k)$ **then**

`// Descent step`

set $\hat{y}^{k+1} \leftarrow \bar{y}^k$, $f^{k+1} \leftarrow \bar{f}^k$

else

`// Null step`

set $\hat{y}^{k+1} \leftarrow \hat{y}^k$, $f^{k+1} \leftarrow f^k$

Form new model $\hat{f}^{k+1}$ so that

$$\hat{f}^{k+1}(y) \geq \max\{\tilde{l}^k + \langle \tilde{g}^k, y \rangle, \bar{l}^k + \langle \bar{g}^k, y \rangle\}$$

with $\tilde{l}^k = \bar{f}^k - \langle \tilde{g}^k, \bar{y}^k \rangle$.

Set $k \leftarrow k + 1$

where

$$g(x) = b - Cx. \tag{5.8}$$

We collect all minorants that can be defined like this in the set

$$W := \{w = (l, x) \in \mathbb{R}^R \times \mathcal{X} \colon l_r = h_r(x_r), r \in R\}$$

and write

$$\hat{f}_w(y) := l + \langle g(x), y\rangle, \quad w = (l, x) \in W.$$

Each convex combination of these minorants is also a minorant of f, *i. e.*

$$\hat{f}_w(y) := l + \langle g(x), y\rangle, \quad w = (l, x) \in \operatorname{conv} W.$$

Note that the function f can be described exactly in terms of these minorants,

$$f(y) = \max\{\hat{f}_w(y) \colon w \in W\} = \max\{\hat{f}_w(y) \colon w \in \operatorname{conv} W\}.$$

In our context the cutting plane models $\hat{f}^k$ will be defined by choosing a finite (compact) number of minorants $\widehat{W}^k \subseteq \operatorname{conv} W$,

$$\widehat{W}^k = \{(l^i, x^i) \in \operatorname{conv} W \colon i \in I^k\},$$

(the difference to the original definition (5.2) is that we use the generating primal $x^k \in \operatorname{conv} \mathcal{X}$ instead of the subgradient g^k, *i. e.* $g^k = g(x^k)$). Given the linear relation (5.8), the aggregated minorant, defined as the unique optimiser of (5.6), can be worked out to be generated by the *aggregated primal* $\bar{x} \in \operatorname{conv} \mathcal{X}$: if $\bar{g} = \sum_{i \in I} \alpha_i g(x^i)$ then $\bar{g} = g(\bar{x})$ with $\bar{x} = \sum_{i \in I} \alpha_i x^i$. Furthermore one can show that each cluster point of an appropriate subsequence of aggregated minorants $(\bar{x}^k)_{k \in \Sigma}$, $\Sigma \subseteq \mathbb{N}$, is an optimal solution of the primal problem (P) if f is bounded, see [31].

Theorem 5.4 *(Theorem 5.2 in [31])*
Suppose $\varepsilon = 0$ and that $f^* := \lim_{k \in \mathbb{N}} f(\hat{y}^k) > -\infty$. Then sequence $(\bar{x}^k)_{k \in \mathbb{N}}$ generated by Algorithm 5.1 is bounded and there is an appropriate subsequence, $(\bar{x}^k)_{k \in \Sigma^*}$, $\Sigma^* \subset \mathbb{N}$, that converges to an optimal solution of (P).

(Note that in our setting the sequence $(\bar{x}^k)_{k \in \mathbb{N}}$ is trivially bounded because the ground set $\mathcal{X}$ is compact. The subsequence converging to an optimal solution is in the case of an infinite number of descent steps the subsequence that corresponds to descent steps, or in the case of a finite number of descent steps the subsequence corresponding to the null steps in the final centre.)

The parallel bundle algorithm to be developed in this chapter will always work in the context of Lagrangian relaxation and use this kind of cutting plane models.

Remark 5.5 In the case of *inequality* constraints the Lagrangian relaxation problem reads

$$\begin{aligned} \text{minimise} \quad & f(y) \\ \text{subject to} \quad & y \in \mathbb{R}^{\mathcal{E}} \times \mathbb{R}^{\mathcal{I}}_{+}, \end{aligned}$$

see Section 3.3. The difference to (LD) are the box constraints $y_{\mathcal{I}} \geq 0$. In order to retain valid dual candidates $\bar{y}$, these constraints must be respected in the augmented subproblem (5.3) as well, see, *e. g.*, [50] for details.

5.2.1 Cutting Planes

When dealing with problems arising from Lagrangian relaxation, one often wants to deal with *primal cutting planes* (here cutting planes do not refer to the same objects as in the "cutting plane model" above, so we will use the term "primal cutting plane" to avoid confusion). Primal cutting planes approaches mean the following. Instead of dealing with the full set of constraints $M = \mathcal{E}$ (resp. $M = \mathcal{E} \cup \mathcal{I}$, we restrict to equality constraints only in the following to simplify notation), one considers in each iteration only a subset of the constraints $M^k \subseteq M$. The next candidate is then determined by the problem

$$\begin{aligned} \text{minimise} \quad & f^k(y) \\ \text{subject to} \quad & y \in \mathbb{R}^{M^k}, \end{aligned}$$

where

$$\begin{aligned} & f^k \colon \mathbb{R}^{M^k} \to \mathbb{R}, \\ & f^k(y) = \langle b_{M^k}, y \rangle + \sum_{r \in R} \max \left\{ h_r(x_r) + \langle -C^T_{M^k,r} y, x_r \rangle \colon x_r \in \mathcal{X}_r \right\}. \end{aligned}$$

In other words, all variables belonging to $M \setminus M^k$ are treated as zero.

Starting with an initial point $\hat{y}^0 = 0$, this problem is equivalent to the original problem as long as none of the components of y in $M \setminus M^k$ is changed. Let $\bar{x}$ be the current primal aggregate. If a constraint $j \in M \setminus M^k$ that is currently not contained in the model is violated by $\bar{x}$, *i. e.* $C_{j,\bullet}\bar{x} \neq b_j$, then by (5.8) the corresponding component of the subgradient is non-zero,

$$g(\bar{x})_j = b_j - C_{j,\bullet} \neq 0.$$

Consequently, although not required in the current centre (because $\hat{y}^k_j = 0$), this constraint may be important for the next point. In fact, one can show that adding the most violated constraint to the model in each model evaluation suffices in order to get a convergent algorithm under reasonable assumptions, see [51]. Furthermore, other separation strategies than adding the most violated constraint may be chosen as well, see [4].

In our practical application we will use primal cutting planes for all constraints with the classical bundle method. The parallel bundle method to be developed in this chapter will not deal with primal cutting planes. However, the extended version of the parallel bundle method (Section 5.4) can be developed into an algorithm that does support primal cutting planes as well, see Remark 5.61.

5.3 Parallel Bundle Method

5.3.1 Introduction

The classical bundle method described in Section 5.2 works in each iteration on the full space of variables $y \in \mathbb{R}^M$ and updates the whole cutting model $\widehat{W} \subset \operatorname{conv} W$. This requires that in each iteration each of the components $f_r(y)$, $r \in R$, of the dual function $f(y) = b^T y + \sum_{r \in R} f_r(y)$ is evaluated. Hence, in each iteration each subproblem (Sub_r), $r \in R$, has to be solved w. r. t. the current candidate point $\bar{y} \in \mathbb{R}^M$. Now suppose the problem contains a large number of subproblems and the coupling between those is relatively loose in the sense that each single coupling constraint $C_{i,\bullet} x = b_i$, $i \in M$, interacts only with a small number of the subproblems. In this case a change of a single component $\hat{y}_i$ would only influence few of the subproblems.

These considerations motivate the following idea of the parallel bundle method. Let $j \in M$ be an arbitrary variable corresponding to a coupling constraint. This specific constraint only interacts (directly) with subproblems $r \in R$ that have non-zero coefficients in this constraint, *i. e.* $R_j := \{r \in R \colon C_{j,r} \neq 0\}$ (remember, $C_{j,r}$ denotes row $j \in M$ of the constraint matrix C with the columns associated with subproblem $r \in R$, so $C_{j,r}$ is a row vector). Note that only the functions $f_r(y)$ with $r \in R_j$ depend on the value of y_j, *i. e.* it holds

$$\forall\, r \notin R_j, \forall\, y, y' \in \mathbb{R}^M, y_i = y'_i, i \neq j \colon f_r(y) = f_r(y'). \tag{5.9}$$

We generalise this notation to larger subspaces of $\mathbb{R}^M$ that correspond to subsets $J \subseteq M$.

Notation Let $r \in R$, $j \in M$, and $J \subseteq M$. Then

J_r	$J_r := \{j \in M \colon C_{j,r} \neq 0\},$	the set of all constraints interacting with r,	(5.10)
R_j	$R_j := \{r \in R \colon j \in J_r\},$	the set of all subproblems interacting with j,	(5.11)
R_J	$R_J := \bigcup_{j \in J} R_j,$	the set of all subproblems interacting with J,	(5.12)
$\bar{J}$	$\bar{J} := \bigcup_{r \in R_J} (J_r \setminus J),$	constraints in $M \setminus J$ interacting with some $r \in R_J$.	(5.13)

The idea is therefore to select a subspace $J \subseteq M$ of constraints and optimise $f(y)$ only on this space. In the view of (5.9) this requires only the evaluation of the functions $f_r(y)$

for $r \in R_J$. Thus the computational effort to optimise $f(y)$ only on the subspace J may be a lot smaller compared with the original problem if R_J is small. We illustrate the structure of the constraint matrix C w. r. t. subspace J and its associated problems R_J in Figure 5.1.

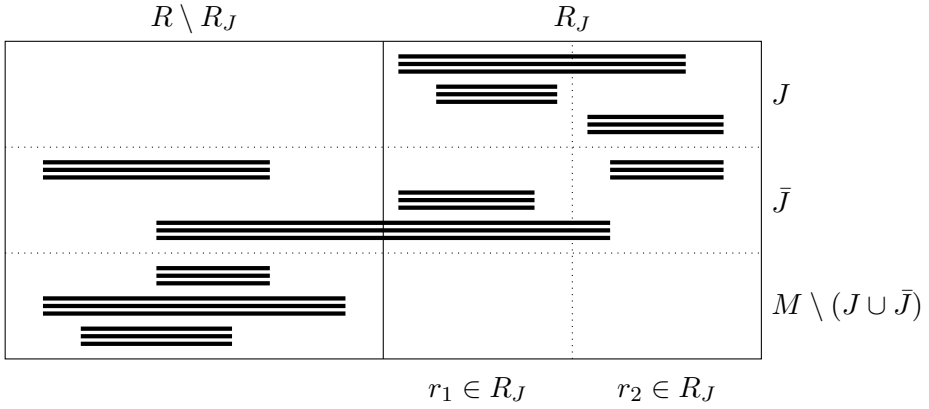


Figure 5.1: The thick lines indicate non-zero coefficients of the constraint matrix. A subspace $J \subseteq M$ divides the subproblems R into two parts: the subproblems R_J, which interact with J, and $R \setminus R_J$, which do not interact with J. The constraints $\bar{J}$ are those that interact with R_J and possibly also $R \setminus R_J$, thus are the coupling between both parts.

The following observation states some immediate consequences of these definitions.

Observation 5.6 *Let $J \subseteq M$ be a subspace. Then there holds*

$$\begin{aligned} &J_r \subseteq J \cup \bar{J}, && \text{for all } r \in R_J, \\ &J_r \cap J = \emptyset, && \text{for all } r \in R \setminus R_J, \\ &R_j \cap R_J = \emptyset, && \text{for all } j \in M \setminus (J \cup \bar{J}). \end{aligned}$$

Proof. Directly from the definitions. □

Now given that a subspace J only influences a small number of problems R_J one may select *simultaneously* another subspace $J' \subseteq M \setminus (J \cup \bar{J})$ so that $R_{J'} \cap R_J = \emptyset$. Those two subspaces do not influence each other and we can optimise $f(y)$ on both subspaces at the same time. In particular, we optimise the $f_r(y)$, $r \in R_J$, by only changing y on $\mathbb{R}^J$ and independently $f_r(y)$, $r \in R_{J'}$, by only changing y on $\mathbb{R}^{J'}$.

In the following sections we develop these ideas into a working parallel and asynchronous bundle algorithm. In Section 5.3.2 we establish the formal framework. Then in Section 5.3.3 we introduce the global structure of the algorithm and how it is divided

into different parallel subprocesses. Each subprocess consists of three major steps, the subspace selection step, the subspace solution step and the update step. We discuss all three steps in order in the sections 5.3.4, 5.3.5 and 5.3.6. Afterwards in Section 5.3.7 we present the basic parallel bundle algorithm. The dependencies between the global problem and the subprocesses are analysed in Section 5.3.8, which finally result in some convergence results presented in Section 5.3.9.

5.3.2 Notation

In order to develop an algorithm that works in parallel by optimising the problem on dynamically chosen subspaces, we must carefully break the problem into its building parts. We also have to state the ingredients of bundle methods (see Section 5.2), *i. e.* the cutting model, w. r. t. certain subspaces and subsets of subproblems. For this we write the original optimisation problem again as

$$\text{(P)} \qquad \begin{aligned} \text{maximise} \quad & h(x) := \sum_{r \in R} h_r(x_r) \\ \text{subject to} \quad & \sum_{r \in R} C_{\bullet,r} x_r = b \\ & x_r \in \mathcal{X}_r, r \in R. \end{aligned}$$

Each $r \in R$ defines a special "Lagrangian function" $L_r(x_r, y) \colon \mathcal{X}_r \times \mathbb{R}^M \to \mathbb{R}$ by

$L_r(x_r, y)$
$$L_r(x_r, y) = h_r(x_r) - y^T C_{\bullet,r} x_r \tag{5.14}$$

and a (dual) function

$f_r(y)$
$$f_r(y) = \max_{x_r \in \mathcal{X}_r} L_r(x_r, y). \tag{5.15}$$

Putting these parts together we get again our original problem.

$$L(x, y) = b^T y + \sum_{r \in R} L_r(x_r, y) \tag{5.16}$$

$$f(y) = \max_{x \in \mathcal{X}} L(x, y) = b^T y + \sum_{r \in R} \max_{x_r \in \mathcal{X}_r} L_r(x_r, y). \tag{5.17}$$

The problem of finding the best Lagrangian relaxation of (P) then reads

$$\text{(LD)} \qquad \begin{aligned} \text{minimise} \quad & f(y) \\ \text{subject to} \quad & y \in \mathbb{R}^M. \end{aligned}$$

Like the classical bundle method, the parallel bundle method will build a cutting plane model of the objective function. This cutting plane model is defined by certain linear minorants of the dual function $f(y)$. Because we will later work only on some $f_r(y)$, $r \in R$, but not all, we need to specify the cutting plane model for each function $f_r(y)$, too.

For $x \in \mathfrak{X}$ the functions $L_r(x_r, \cdot)$, $r \in R$, and $L(x, \cdot)$ are linear in the second argument with gradients

$$g_r(x_r) := -C_{\bullet,r}x_r, r \in R, \qquad \text{and} \qquad g(x) = b - Cx = b + \sum_{r\in R} g_r(x_r), \tag{5.18}$$

and we define

$$\begin{aligned} W_r &:= \{w_r = (l_r, x_r) \in \mathbb{R} \times \mathfrak{X}_r \colon l_r = h_r(x_r)\}, \\ W &:= \bigtimes_{r\in R} W_r. \end{aligned} \tag{5.19}$$

Each point $w_r = (l_r, x_r) \in \operatorname{conv} W_r$ defines a linear minorant of f_r, $r \in R$,

$$\hat{f}_{w_r,r}(y) := l_r + g_r(x_r)^T y \le f_r(y) \tag{5.20}$$

and putting them together we get a minorant of f

$$\hat{f}_w(y) := \sum_{r\in R} l_r + g(x)^T y = b^T y + \sum_{r\in R} \hat{f}_{w_r,r}(y) \le f(y).$$

As before, if we use a set $R' \subseteq R$ instead of a subscript r we sum up the single values, *e. g.*,

$$f_{R'}(y) := \sum_{r\in R'} f_r(y). \tag{$f_{R'}(y)$}$$

Notation Elements of the minorant sets $\operatorname{conv} W$ or $\operatorname{conv} W_r$ will *always* be denoted by a letter "w". We will always use the convention that w consists of two parts, *i. e.* $w = (l, x)$ with $l \in \mathbb{R}^R$ denoting the first and $x \in \operatorname{conv} \mathfrak{X}$ the second part. When w is decorated or has a subscript, the l and x with the same decoration and subscript denote the respective parts of w, *e. g.* if $\bar{w}_r \in \operatorname{conv} W_r$ then $\bar{l}_r \in \mathbb{R}$ and $\bar{x}_r \in \operatorname{conv} \mathfrak{X}_r$ with $\bar{w}_r = (\bar{l}_r, \bar{w}_r)$.

The following simple observation follows directly from the definitions above and states some fundamental relations for the single parts.

Observation 5.7 *The following relations hold.*

$$\forall\, r \in R, \forall\, j \in M\colon \quad j \in J_r \iff r \in R_j, \tag{5.21}$$

$$\forall\, r \in R, \forall\, y \in \mathbb{R}^M\colon \quad (C_{J_r,r})^T y_{J_r} = (C_{\bullet,r})^T y, \tag{5.22}$$

$$\forall\, r \in R, \forall\, y, y' \in \mathbb{R}^M\colon \quad y_{J_r} = y'_{J_r} \;\Rightarrow\; f_r(y) = f_r(y'), \tag{5.23}$$

$$\forall\, r \in R, \forall\, w_r \in \operatorname{conv} W_r, \forall\, y, y' \in \mathbb{R}^M\colon \quad y_{J_r} = y'_{J_r} \;\Rightarrow\; \hat{f}_{w_r,r}(y) = \hat{f}_{w_r,r}(y'), \tag{5.24}$$

$$\forall\, J \subseteq M, \forall\, x, x' \in \mathbb{R}^n\colon \quad x_{R_J} = x'_{R_J} \;\Rightarrow\; g(x)_J = g(x')_J. \tag{5.25}$$

Proof. Directly from the definitions. □

A fundamental quantity for bundle methods is the expected progress or expected decrease (cf. (5.7)) $\Delta(\bar{w}, \hat{y}) = f(\hat{y}) - \hat{f}_{\bar{w}}(\hat{y}) + \frac{1}{u}\|g(\bar{x})\|^2$ where $\hat{y} \in \mathbb{R}^M$ denotes the current centre of stability and $\bar{w} = (\bar{l}, \bar{x}) \in \operatorname{conv} W$ denotes the current primal aggregate. Similar to the functions above we split up this value w. r. t. a certain subspace $J \subseteq M$ as follows with $\bar{w} = (\bar{l}, \bar{x}) \in \operatorname{conv} W$

$$\Delta(\bar{w}, \hat{y}) = \sum_{r \in R} \left[f_r(\hat{y}) - \hat{f}_{\bar{w}_r, r}(\hat{y}) \right] + \frac{1}{u}\|g(\bar{x})\|^2, \tag{5.26}$$

$$\Delta_J(\bar{w}, \hat{y}) := \sum_{r \in R_J} \left[f_r(\hat{y}) - \hat{f}_{\bar{w}_r, r}(\hat{y}) \right] + \frac{1}{u}\|g(\bar{x})_J\|^2, \tag{5.27}$$

$$\bar{\Delta}_J(\bar{w}, \hat{y}) := \sum_{r \in R \setminus R_J} \left[f_r(\hat{y}) - \hat{f}_{\bar{w}_r, r}(\hat{y}) \right] + \frac{1}{u}\|g(\bar{x})_{M \setminus (J \cup \bar{J})}\|^2, \tag{5.28}$$

$$\delta_{\bar{J}}(\bar{w}) := \frac{1}{u}\|g(\bar{x}_r)_{\bar{J}}\|^2.$$

These values can be interpreted from the perspective of the subspace J and its associated subproblems R_J. Given a current centre of stability $\hat{y} \in \mathbb{R}^M$ and a primal aggregate $\bar{w} = (\bar{l}, \bar{x}) \in \operatorname{conv} W$ the value $\Delta_J(\bar{w}, \hat{y})$ is the predicted decrease if only the variables $\hat{y}_J$ along with the primal variables $\bar{x}_{R_J}$ are allowed to change and all other variables are fixed. Similarly the value $\bar{\Delta}_J(\bar{w}, \hat{y})$ is the part of the predicted decrease that cannot be influenced by only changing $\bar{x}_{R_J}$ or $\hat{y}_J$. Finally $\delta_{\bar{J}}(\bar{w})$ is the residual part that can be influenced by both, the subproblems R_J and the subproblems not interacting with J, namely $R \setminus R_J$. The following observation justifies this "splitting" of the predicted decrease.

Observation 5.8 *Let $J \subseteq M$ be a subspace and let $\bar{w} = (\bar{l}, \bar{x}) \in \operatorname{conv} W$ and $\hat{y} \in \mathbb{R}^M$. Then*

$$\Delta(\bar{w}, \hat{y}) = \Delta_J(\bar{w}, \hat{y}) + \bar{\Delta}_J(\bar{w}, \hat{y}) + \delta_{\bar{J}}(\bar{w}).$$

Proof. Direct computation. □

5.3.3 The Parallel Algorithmic Framework

This section gives a rough sketch of the parallel bundle method. The purpose is to introduce how parallelism works in our case and how we denote the different parallel subprocesses and their associated "local" data as well as the shared "global" data.

The most important aspect of the algorithm for this section is the handling of a global index marker $\sigma \in \mathbb{N}_0$ as well as of two local index markers $\underline{\pi}, \bar{\pi} \in \mathbb{N}_0 \cup \{-1, \infty\}$. We

will see that the global index marker can be associated with a unique sequence of states of the global data, and the local index markers with the states when a certain parallel subprocess starts and ends. We leave out all technical details and refer to later sections for them. The outline of the algorithm is as follows.

1. **Initialisation:** Initialisation of global data.
   Set $\sigma \leftarrow 0$

2. **Parallel optimisation:** Start parallel subprocess. The indexes $\underline{\pi}, \bar{\pi} \in \mathbb{N}_0 \cup \{-1, \infty\}$ are *local* data for each process. Those are initialised to $\underline{\pi} = \bar{\pi} = -1$. Each process does the following.

   a) **Subspace selection:**

      - Acquire *exclusive* access to the global data.
      - $\underline{\pi} \leftarrow \sigma$,
      - Select an appropriate subspace for optimisation. If no appropriate subspace could be selected, STOP this process.
      - Otherwise
        - get and block the global subspace data,
        - $\sigma \leftarrow \sigma + 1$,
        - $\bar{\pi} \leftarrow \infty$.
      - Free exclusive access.

   b) **Subspace optimisation:** Without any global interaction, solve the subspace problem until some termination criterion is satisfied.

   c) **Subspace update:**

      - Acquire *exclusive* access to the global data.
      - $\bar{\pi} \leftarrow \sigma$ and $\sigma \leftarrow \sigma + 1$.
      - Update the global data on the subspace with the optimised values.
      - If some global stopping criterion is satisfied, STOP the whole algorithm.
      - Free exclusive access.
      - STOP this process.

The main process first initialises the global data and then starts several parallel subprocesses. Each single subprocess does up to three steps in order: subspace selection a), subspace optimisation b) and subspace update c). Note that a process does not necessarily complete each step. The first step, subspace selection, may fail and in this case the process stops immediately. The other steps cannot fail, but the whole algorithm may stop while some process is still running. This may happen because several processes run in parallel: while one process is still optimising, another process may finish and update the global data. If then the global data satisfies some stopping criterion *all* processes

stop, no matter whether they are still processing their respective subspace problem or not. But only processes that complete at least the first step actually influence the global solution. Such processes are called *relevant*, and we will usually deal only with relevant processes. In other words, processes that do not successfully select a subspace will be ignored completely because they do not change any global data at all.

relevant

The only interaction with, and particularly the only change of, global data happens in the first and the third step. As in any parallel framework, simultaneous writing (or reading and writing) to the (shared) global data from more than one parallel subprocess must be forbidden using some locking mechanism. Because of this there is a *unique sequence* of the interactions of the processes with the global data. Thus we can assign each state of the global data a unique index $\sigma \in \mathbb{N}_0$. Each global object will be denoted by a superscript σ in angle braces $\langle\sigma\rangle$. For example, the algorithm will manage a global centre of stability $\hat{y} \in \mathbb{R}^M$. The global centre with index σ will be denoted by $\hat{y}^{\langle\sigma\rangle}$.

$\langle\sigma\rangle$

Let π denote some specific subprocess. The uniqueness of the global index allows to equip each subprocess with the two special index markers $\underline{\pi}$ and $\bar{\pi}$ that correspond to the global state when the process starts its subspace selection step and its subspace update step, respectively. It will be convenient to identify a process and its state with this pair of index markers. When subprocess π is initialised we set both markers to $\underline{\pi} = -1 = \bar{\pi}$. Whenever the process tries to acquire a subspace in step a) the label $\underline{\pi}$ is set to the current value of σ. If the subprocess succeeds in selecting the subspace this will be the final value of $\underline{\pi}$ and the new status is indicated by setting $\bar{\pi} \leftarrow \infty$ and σ is increased. Once the process has completed its subspace problem and enters the update step c) the label $\bar{\pi}$ is assigned the new current value of σ as its final value and σ will be increased again. In consequence, a process π with $\bar{\pi} \leq \underline{\pi}$ is waiting for a suitable subspace, a process with $\bar{\pi} = \infty$ is currently working on its subproblem, and if $\underline{\pi} < \bar{\pi} < \infty$ the process has done its work. Because each process that successfully executes its subspace selection step can be uniquely identified by these two markers we will often write $\pi = (\underline{\pi}, \bar{\pi})$.

$\underline{\pi}$, $\bar{\pi}$

Remark 5.9 When describing the algorithm it will be convenient to think of each subprocess as being terminated once it has completed the processing of its subspace problem. This has the advantage that there is a one-to-one correspondence between processes and subspace problems. In practice, however, one would not terminate a subprocess. Instead it would simply try to acquire a new subspace problem.

Furthermore, if the subspace selection step fails, then it makes no sense to start another process immediately before some already running process has stopped. This is because the global data would not change as long as no running process stops, thus a new attempt to select a subspace would use the same global data and fail, too. Note that we will have to ensure that the subspace selection never fails if no process is currently running (see Observation 5.14).

For the analysis it will be convenient to arrange the processes in different groups for

each global index $\sigma \in \mathbb{N}_0 \cup \{\infty\}$:

$$\underline{\Pi}^{\langle\sigma\rangle} := \{\pi = (\underline{\pi}, \bar{\pi}) \colon \underline{\pi} < \bar{\pi} \text{ and } \underline{\pi} < \sigma\}, \qquad \underline{\Pi}^{\langle\sigma\rangle}$$

$$\bar{\Pi}^{\langle\sigma\rangle} := \{\pi = (\underline{\pi}, \bar{\pi}) \colon \underline{\pi} < \bar{\pi} < \sigma\}, \qquad \bar{\Pi}^{\langle\sigma\rangle}$$

$$\Pi^{\langle\sigma\rangle} := \{\pi = (\underline{\pi}, \bar{\pi}) \colon \underline{\pi} < \sigma \leq \bar{\pi}\} = \underline{\Pi}^{\langle\sigma\rangle} \setminus \bar{\Pi}^{\langle\sigma\rangle}. \qquad \Pi^{\langle\sigma\rangle}$$

The first group $\underline{\Pi}^{\langle\sigma\rangle}$ collects all processes that have successfully executed step a) before the algorithm reached σ, $\bar{\Pi}^{\langle\sigma\rangle}$ singles out all processes that have executed c) before σ and the set $\Pi^{\langle\sigma\rangle}$ comprises all processes that are actively working on or just finishing a subproblem at σ, *i. e.*, these are running in parallel. Note that the set $\underline{\Pi}^{\langle\infty\rangle}$ is precisely the set of all relevant processes. Each increment of σ by the algorithm is associated with the addition or deletion of exactly one process from the set $\Pi^{\langle\sigma\rangle}$ of parallel processes as we show next.

Observation 5.10 *The following relations hold.*

(i) $\Pi^{\langle 0\rangle} = \emptyset$,

(ii) for $\sigma' \in \mathbb{N}_0$ *there holds* $\Pi^{\langle\sigma'\rangle} \neq \Pi^{\langle\sigma'+1\rangle}$ *if and only if*

$$|(\Pi^{\langle\sigma'\rangle} \setminus \Pi^{\langle\sigma'+1\rangle}) \cup (\Pi^{\langle\sigma'+1\rangle} \setminus \Pi^{\langle\sigma'\rangle})| = 1,$$

(iii) if $\Pi^{\langle\sigma'\rangle} = \Pi^{\langle\sigma'+1\rangle}$ *then for all* $\sigma \geq \sigma'$ *we have* $\Pi^{\langle\sigma\rangle} = \Pi^{\langle\sigma'\rangle}$ *and there is no process* π *with* $\underline{\pi} = \sigma$ *or* $\bar{\pi} = \sigma$.

Proof. The first statement $\Pi^{\langle 0\rangle} = \emptyset$ follows by definition. For the second statement $|(\Pi^{\langle\sigma'\rangle} \setminus \Pi^{\langle\sigma'+1\rangle}) \cup (\Pi^{\langle\sigma'+1\rangle} \setminus \Pi^{\langle\sigma'\rangle})| = 1 \Rightarrow \Pi^{\langle\sigma'\rangle} \neq \Pi^{\langle\sigma'+1\rangle}$ is obvious. In order to show the other implication observe that $\Pi^{\langle\sigma'\rangle} \neq \Pi^{\langle\sigma'+1\rangle}$ implies $|(\Pi^{\langle\sigma'\rangle} \setminus \Pi^{\langle\sigma'+1\rangle}) \cup (\Pi^{\langle\sigma'+1\rangle} \setminus \Pi^{\langle\sigma'\rangle})| \geq 1$, so there must be at least one process π that is in $\Pi^{\langle\sigma'\rangle}$ but not in $\Pi^{\langle\sigma'+1\rangle}$, *i. e.*, it satisfies $\bar{\pi} = \sigma'$, or that is in $\Pi^{\langle\sigma'+1\rangle}$ but not in $\Pi^{\langle\sigma'\rangle}$, *i. e.*, it satisfies $\underline{\pi} = \sigma'$. Due to the exclusive access in steps a) and c) there is exactly one such process π with $\bar{\pi} \geq 0$. Indeed, any process $\pi \in \Pi^{\langle\sigma'+1\rangle}$ with $\underline{\pi} = \sigma'$ executed step a) successfully at $\sigma = \sigma'$ (unsuccessful steps have $\bar{\pi} < 0$) and increased the marker to $\sigma = \sigma' + 1$ so that all further executions of steps a) and c) lead to larger numbers. An analogous argument holds for the case $\bar{\pi} = \sigma'$.

If $\Pi^{\langle\sigma'\rangle} = \Pi^{\langle\sigma'+1\rangle}$ then no process $\pi \in \Pi^{\langle\sigma'\rangle}$ executes a) at $\sigma = \sigma'$ and there is no new successful execution of c) at $\sigma = \sigma'$. Thus, σ is never increased above $\sigma' + 1$, all running processes $\pi \in \Pi^{\langle\sigma'\rangle}$ satisfy $\bar{\pi} = \infty$, all others have $\bar{\pi} < \sigma' + 1$ and none of these values ever change, so the claim follows. □

By this observation, the following set collects the markers $\sigma \in \mathbb{N}_0$ visited by the algorithm,

$$\Sigma := \{0\} \cup \{\sigma \in \mathbb{N} \colon \Pi^{\langle\sigma\rangle} \neq \Pi^{\langle\sigma-1\rangle}\}. \qquad \Sigma$$

5.3.4 Subspace Selection

The first important step in the parallel bundle method is the selection of appropriate subspaces on which the subprocesses should work. Those subspaces must be selected carefully so that the processes, when running in parallel, influence each other only in a predictable way.

Optimising a subspace J implies that the process must deal with the subproblems R_J. Hence, when the process optimising J and R_J is running, no other process is allowed to deal with any $r \in R_J$ at the same time. The algorithm will later keep track of a set of *blocked subproblems* $B \subseteq R$ that contains all subproblems that are currently processed by some subprocess. The first statement in this section states simple relations between two subspaces whose associated subproblems do not intersect.

Observation 5.11 *Let J, $J' \subseteq M$ be two subspaces with $R_J \cap R_{J'} = \emptyset$. Then*

$$J \cap (J' \cup \bar{J}') = J' \cap (J \cup \bar{J}) = \emptyset.$$

Proof. Assume for a contradiction that there exists w. l. o. g. $j \in J \cap (J' \cup \bar{J}')$. Assumption (5.1) implies $R_j \neq \emptyset$. Because $j \in J$ we know by definition (5.12) that $R_j \subseteq R_J$. If $j \in J'$ then analogously $R_j \subseteq R_{J'}$, contradicting $\emptyset \neq R_j \subseteq R_J \cap R_{J'} = \emptyset$. So we may assume $j \in \bar{J}'$. But then there must be an $r \in R_{J'}$ with $j \in J_r$ and therefore $r \in R_j$. This implies $r \in R_j \cap R_{J'} \subseteq R_J \cap R_{J'}$, a contradiction. □

Observation 5.11 implies that it is sufficient to ensure that the parallel processes work on subspaces influencing disjoint sets of subproblems. In this case the associated subspaces are automatically disjoint, too.

The subspace to be selected must provide a sufficiently large progress compared with the current global expected progress. This means, given a current centre of stability $\hat{y} \in \mathbb{R}^M$ and an aggregate minorant $\bar{w} \in \operatorname{conv} W$ some $\tau_1 \in (0, 1]$, the subspace $J \subseteq M$ must satisfy

$$\Delta_J(\bar{w}, \hat{y}) \geq \tau_1 \Delta(\bar{w}, \hat{y}). \tag{5.29}$$

The parameter τ_1 determines how large the expected subspace progress must be in comparison to the global expected progress. The following simple observation verifies that this condition can always be met by selecting a sufficiently large subspace.

Observation 5.12 *For subspaces $J \subseteq J' \subseteq M$ and $\bar{w} \in \operatorname{conv} W$, $\hat{y} \in \mathbb{R}^M$ there holds*

$$\Delta_J(\bar{w}, \hat{y}) \leq \Delta_{J'}(\bar{w}, \hat{y}) \tag{5.30}$$

Furthermore, condition (5.29) holds for $J = M$.

Proof. The first assertion follows directly from definition (5.27) because $f_r \geq \hat{f}_{r,\bar{w}}$. The second assertion follows because $R_M = R$, thus $\Delta_M(\bar{w}, \hat{y}) = \Delta(\bar{w}, \hat{y})$ by Observation 5.8. □

These observations imply that a simple greedy strategy, that increases the subspace to be selected successively, will always succeed. But usually we are interested in rather small subspaces that require less computational effort and allow for a larger number of other non-blocked subspaces. Choosing τ_1 small enough guarantees that there is always a one-dimensional subspace that fulfils (5.29).

Lemma 5.13 *If $\tau_1 \in (0, \frac{1}{2}\min\{\frac{1}{|M|}, \frac{1}{|R|}\})$ then there is always a $j \in M$ so that $J = \{j\}$ fulfils (5.29).*

Proof. Recall the definition (5.26) of $\Delta(\bar{w}, \hat{y})$

$$\Delta(\bar{w}, \hat{y}) = \sum_{r \in R} \left[f_r(\hat{y}) - \hat{f}_{\bar{w}_r, r}(\hat{y}) \right] + \frac{1}{u} \|g(\bar{x})\|^2.$$

At least one of both summands is greater than or equal to $\frac{1}{2}\Delta(\bar{w}, \hat{y})$. If this is true for the first one, then for at least one $r \in R$ the term $f_r(\hat{y}) - \hat{f}_{\bar{w}_r, r}(\hat{y})$ is greater than or equal to $\frac{1}{2}\Delta(\bar{w}, \hat{y})\frac{1}{|R|}$ and any $j \in J_r$ satisfies the claim. If it is true for the second one, then for at least one $j \in M$ the term $\frac{1}{u} g(\bar{x})_j^2$ is greater than or equal to $\frac{1}{2}\Delta(\bar{w}, \hat{y})\frac{1}{|M|}$. □

In the following we will always assume τ_1 to fulfil the requirements of Lemma 5.13.

An important aspect of the parallel bundle algorithm is the automatic detection of dependencies between different constraints. We will discuss this detection in Section 5.3.6. However, once a dependency between certain constraints is detected, it has to be respected in future subspace selections.

A dependency between two constraints $j, j' \in M$, $j \neq j'$, is of the following form: whenever constraint j is selected for a subspace J, the constraint j' must be included in the subspace as well. Thus, we model the dependencies by a directed *dependency graph* $D = (M, E)$ with the constraints being the nodes and set of arcs $D = (M, E)$

$$E \subseteq \{(j, j') \in M \times M : j \neq j'\}.$$

An arc $(j, j') \in E$ corresponds exactly to a dependency as described above, *i. e.* the subspace selection must ensure

$$j \in J \wedge (j, j') \in E \Rightarrow j' \in J.$$

Note that if the dependency graph D is complete, the only valid subspace is the whole space $J = M$. At the beginning of the algorithm the dependency graph D will contain no arcs. New arcs will be added when new dependencies are detected, but no arc will ever

be removed. Hence, the set of arcs will grow during the algorithm and we will denote the graph associated with a certain global index by $D^{\langle\sigma\rangle} = (M, E^{\langle\sigma\rangle})$.

Similarly, as stated before, we have to ensure that the subspace problem of each two processes that run in parallel work on disjoint subspaces resp. sets of subproblems. We
$B^{\langle\sigma\rangle}$ do this by keeping track of the set of blocked subproblems $B^{\langle\sigma\rangle}$ at index $\sigma \in \mathbb{N}_0$. This set contains the subproblems of all subspace problems that start before σ and stop not before σ, *i. e.*

$$B^{\langle\sigma\rangle} = \bigcup_{r \in \Pi^{\langle\sigma\rangle}} R_{J^{(\pi)}}.$$

When the algorithm starts no subproblem is blocked, *i. e.* $B^{\langle 0\rangle} = \emptyset$.

We can now state a procedure that selects an appropriate subspace w. r. t. a set of blocked subproblems $B^{\langle\sigma\rangle} \subseteq R$ and a dependency graph $D^{\langle\sigma\rangle} = (M, E^{\langle\sigma\rangle})$. The global index σ will always correspond to a start-index $\underline{\pi}$ associated with a process π. This procedure (see Algorithm 5.2) is a simple greedy algorithm that successively adds further constraints to the subspace $J^{(\pi)}$ until condition (5.29) is satisfied.

Algorithm 5.2: SelectSubspace

Input : global data at $\underline{\pi}$
Output : **TRUE** if a subspace can be selected
Changes: sets $J^{(\pi)}$
$X \leftarrow M \setminus \left(\bigcup_{r \in B^{\langle\underline{\pi}\rangle}} J_r\right)$, $J^{(\pi)} \leftarrow \emptyset$
while $J^{(\pi)}$ *does not satisfy (5.29) and* $X \neq \emptyset$ **do**
  select $j \in \operatorname{Argmax}\{\Delta_{\{j\}}(\bar{w}^{\langle\underline{\pi}\rangle}, \hat{y}^{\langle\underline{\pi}\rangle}) \colon j \in X\}$
  $Y \leftarrow \{j\} \cup \{j' \in M \colon (j, j') \in E^{\langle\underline{\pi}\rangle}\}$
  if $R_Y \cap B^{\langle\underline{\pi}\rangle} = \emptyset$ **then**
1 $J^{(\pi)} \leftarrow J^{(\pi)} \cup Y$, $X \leftarrow X \setminus Y$
  else
    $X \leftarrow X \setminus \{j\}$
if $J^{(\pi)}$ *satisfies (5.29)* **then**
  return ***TRUE***
else
  $J^{(\pi)} \leftarrow \emptyset$
  return ***FALSE***

Observation 5.14 *If Algorithm 5.2 returns a subspace* $\emptyset \neq J^{(\pi)} \subseteq M$*, then* $J^{(\pi)}$ *fulfils (5.29) and*

$$R_{J^{(\pi)}} \cap B^{\langle\underline{\pi}\rangle} = \emptyset. \tag{5.31}$$

If the algorithm is called without blocked subproblems, i. e. $B^{\langle\underline{\pi}\rangle} = \emptyset$*, then the returned subspace is not empty, i. e.* $J^{(\pi)} \neq \emptyset$.

Proof. The algorithm is finite because at least one element is removed from X in each iteration. The first statement follows from the final test. The second statement follows inductively: $J^{(\pi)} = \emptyset$ fulfils (5.31) trivially. If $J^{(\pi)}$ is enlarged in A5.2.1, then $R_{J^{(\pi)} \cup Y} \cap B^{\langle \underline{\pi} \rangle} = (R_{J^{(\pi)}} \cap B^{\langle \underline{\pi} \rangle}) \cup (R_Y \cap B^{\langle \underline{\pi} \rangle}) = \emptyset$. The first term is the empty set by induction hypothesis, the latter because of the previous test.

The last statement follows because if $B^{\langle \underline{\pi} \rangle} = \emptyset$ then $J^{(\pi)}$ will eventually become M, which fulfils (5.29) by Observation 5.12. □

5.3.5 The Subspace Problem

In this section we analyse the problem to be solved on a subspace $J \subseteq M$. A subspace problem consists of optimising $f(y)$ by changing y on a certain subspace $J \subseteq M$ while keeping the other values on $M \setminus J$ fixed.

The subspace problem consists again of several parts, each associated with one subproblem $r \in R_J$. We will introduce a notation for the single parts similar to the global problem in Section 5.3.2. Afterwards we build the subspace problem by putting these parts together. As we mentioned in Section 5.3.3 there is a one-to-one correspondence between a subspace problem and the subprocess that solves it. Thus in the following we will denote a subspace problem by its associated subprocess π. In order to distinguish the objects used for the subproblem we will equip them with a superscript π. The data that defines a subspace problem is

(i) the associated subspace $J^{(\pi)} \subseteq M$,

(ii) primal aggregate minorant when the process starts $\bar{w}^{\langle \underline{\pi} \rangle} \in \operatorname{conv} W$

(iii) initial centre of stability when the process starts $\hat{y}^{\langle \underline{\pi} \rangle} \in \mathbb{R}^M$.

We start with the description of the subspace problem by incorporating the influence of the fixed part of y into the objective function

$$c_r^{(\pi)} := -(C_{\bar{J}^{(\pi)},r})^T \hat{y}^{\langle \underline{\pi} \rangle}_{\bar{J}^{(\pi)}}, \qquad \text{for all } r \in R_{J^{(\pi)}}, \tag{5.32}$$

$$h_r^{(\pi)}(x_r) := h_r(x_r) + (c_r^{(\pi)})^T x_r, \qquad \text{for all } x_r \in \mathcal{X}_r, r \in R_{J^{(\pi)}}. \tag{5.33}$$

Note that by definition (5.13) of $\bar{J}$ relation (5.22) implies $c_r^{(\pi)} = -(C_{M \setminus J^{(\pi)},r})^T \hat{y}^{\langle \underline{\pi} \rangle}_{M \setminus J^{(\pi)}}$ for all $r \in R_{J^{(\pi)}}$. Now analogously to Section 5.3.2 we define the Lagrangian and the dual functions

$$L_r^{(\pi)}(x_r, y^{(\pi)}) := h_r^{(\pi)}(x_r) - (y^{(\pi)})^T C_{J^{(\pi)},r} x_r, \quad x_r \in \mathcal{X}_r, r \in R_{J^{(\pi)}}, y^{(\pi)} \in \mathbb{R}^{J^{(\pi)}}, \tag{5.34}$$

$$f_r^{(\pi)}(y^{(\pi)}) := \max_{x_r \in \mathcal{X}_r} L_r^{(\pi)}(x_r, y^{(\pi)}), \qquad r \in R_{J^{(\pi)}}, y^{(\pi)} \in \mathbb{R}^{J^{(\pi)}},$$

$$f^{(\pi)}(y^{(\pi)}) := b^T_{J^{(\pi)}} y^{(\pi)} + \sum_{r \in R_{J^{(\pi)}}} f_r^{(\pi)}(y^{(\pi)}). \tag{5.35}$$

The subspace problem then reads

$$\text{(Sub}(\pi)\text{)} \qquad \begin{aligned} \text{minimise} \quad & f^{(\pi)}(y^{(\pi)}) \\ \text{subject to} \quad & y^{(\pi)} \in \mathbb{R}^J. \end{aligned}$$

The subspace problem will be solved again by a (classical) bundle method. For this we introduce again linear minorants and cutting plane models for the objective functions $f_r^{(\pi)}(y^{(\pi)})$, $r \in R_{J^{(\pi)}}$, and $f^{(\pi)}(y^{(\pi)})$. We collect all feasible primal solutions in

$W^{(\pi)}$

$$W^{(\pi)} := \bigtimes_{r \in R_{J^{(\pi)}}} W_r.$$

Each point $w = (l, x) \in \operatorname{conv} W^{(\pi)}$ defines a minorant of $f^{(\pi)}(y^{(\pi)})$,

$$\begin{aligned} \hat{f}^{(\pi)}_{w_r,r}(y^{(\pi)}) &:= l_r + (c_r^{(\pi)})^T x_r - (y^{(\pi)})^T C_{J^{(\pi)},r} x_r \le f_r^{(\pi)}(y^{(\pi)}), \qquad r \in R_{J^{(\pi)}}, \\ \hat{f}^{(\pi)}_{w}(y^{(\pi)}) &:= b^T_{J^{(\pi)}} y^{(\pi)} + \sum_{r \in R_{J^{(\pi)}}} \hat{f}^{(\pi)}_{w_r,r}(y^{(\pi)}) \le f^{(\pi)}(y^{(\pi)}). \end{aligned} \tag{5.36}$$

The gradient of $\hat{f}^{(\pi)}_{w^{(\pi)}}$ can be worked out to be

$$g^{(\pi)}(x^{(\pi)}) := b_{J^{(\pi)}} - C_{J^{(\pi)},R_{J^{(\pi)}}} x^{(\pi)}. \tag{5.37}$$

The difference between the subspace objects $L_r^{(\pi)}(x_r, y^{(\pi)})$, $\hat{f}^{(\pi)}(y^{(\pi)})$, $f^{(\pi)}(y^{(\pi)})$ and their global counterparts is the additional term $(c_r^{(\pi)})^T x_r$ in the objective functions $h^{(\pi)}(x_r)$. Therefore we get some direct relations between the subspace and global variants.

Observation 5.15 *Let π be a subspace problem with subspace $J^{(\pi)} \subseteq M$. Then the following relations hold for each $y \in \mathbb{R}^M$ with $y_{\bar{J}^{(\pi)}} = \hat{y}^{\langle\pi\rangle}_{\bar{J}^{(\pi)}}$.*

$$L_r^{(\pi)}(x_r, y^{(\pi)}) = L_r(x_r, y), \qquad \text{for all } x_r \in \mathfrak{X}, r \in R_{J^{(\pi)}}, \tag{5.38}$$

$$f_r^{(\pi)}(y^{(\pi)}) = f_r(y), \qquad \text{for all } r \in R_{J^{(\pi)}}, \tag{5.39}$$

$$\hat{f}^{(\pi)}_{w_r,r}(y^{(\pi)}) = \hat{f}_{w_r,r}(y), \qquad \text{for all } w_r \in \operatorname{conv} W_r, r \in R_{J^{(\pi)}}, \tag{5.40}$$

$$g^{(\pi)}(x_{R_{J^{(\pi)}}}) = g(x)_{J^{(\pi)}}, \qquad \text{for all } x \in \operatorname{conv} \mathfrak{X}. \tag{5.41}$$

Proof. We know, for $r \in R$, by Observation 5.6 that $J_r \subseteq J^{(\pi)} \cup \bar{J}^{(\pi)}$ and by (5.10) $C_{M \setminus J^{(\pi)},r} = 0$, so

$$\begin{aligned} c_r^{(\pi)} - C^T_{J^{(\pi)},r} y_{J^{(\pi)}} &\overset{(5.32)}{=} -C^T_{\bar{J}^{(\pi)},r} \hat{y}^{\langle\pi\rangle}_{\bar{J}^{(\pi)}} - C^T_{J^{(\pi)},r} y_{J^{(\pi)}} \\ &= -C^T_{\bar{J}^{(\pi)},r} y_{\bar{J}^{(\pi)}} - C^T_{J^{(\pi)},r} y_{J^{(\pi)}} = -C^T_{\bullet,r} y. \end{aligned}$$

Thus for $x_r \in \mathcal{X}$ we get

$$\begin{aligned} L_r^{(\pi)}(x_r, y_{J^{(\pi)}}) &\overset{(5.33)}{=} h_r(x_r) + (c_r^{(\pi)})^T x_r - (y_{J^{(\pi)}})^T C_{J^{(\pi)},r} x_r \\ &= h_r(x_r) - y^T C_{\bullet,r} x_r \overset{(5.14)}{=} L_r(x_r, y). \end{aligned}$$

This proves (5.38) and also (5.39) by their definitions (5.15) and (5.35). For (5.40) with $w_r = (l_r, x_r)$

$$\begin{aligned} \hat{f}^{(\pi)}_{w_r,r}(y_{J^{(\pi)}}) &\overset{(5.36)}{=} l_r + (c_r^{(\pi)})^T x_r - (y_{J^{(\pi)}})^T C_{J^{(\pi)},r} x_r \\ &= l_r - y^T C_{\bullet,r} x_r \overset{(5.18),(5.20)}{=} \hat{f}_{w_r,r}(y). \end{aligned}$$

Finally, (5.41) relies on $C_{j,R\setminus R_j} = 0$ and $R_j \subseteq R_{J^{(\pi)}}$ for $j \in J^{(\pi)}$ by definition,

$$\begin{aligned} g^{(\pi)}(x^{(\pi)}) &\overset{(5.37)}{=} b_{J^{(\pi)}} - C_{J^{(\pi)},R_{J^{(\pi)}}} x^{(\pi)} - \underbrace{C_{J^{(\pi)},R\setminus R_{J^{(\pi)}}}}_{=0} x_{R\setminus R_{J^{(\pi)}}} \\ &= b_{J^{(\pi)}} - C_{J^{(\pi)},\bullet} x \overset{(5.18)}{=} g(x)_{J^{(\pi)}}. \end{aligned}$$

□

The bundle method solving the subspace problem works analogously to the classical bundle method presented in Section 5.2 applied to (Sub(π)). Given a centre of stability $\hat{y}^{(\pi)} \in \mathbb{R}^{J^{(\pi)}}$ and an aggregate minorant $\bar{w}^{(\pi)} = (\bar{l}^{(\pi)}, \bar{x}^{(\pi)}) \in \operatorname{conv} W^{(\pi)}$, the next candidate point is

$$\bar{y}^{(\pi)} := \hat{y}^{(\pi)} - \frac{1}{u} g^{(\pi)}(\bar{x}^{(\pi)}).$$

The candidate is the minimiser of the augmented model

$$\bar{y}^{(\pi)} = \operatorname{argmin} \left\{ \hat{f}^{(\pi)}_{\bar{w}^{(\pi)}}(y^{(\pi)}) + \frac{u}{2} \|y^{(\pi)} - \hat{y}^{(\pi)}\|^2 \colon y^{(\pi)} \in \mathbb{R}^{J^{(\pi)}} \right\}.$$

The algorithm performs a descent step if the actual decrease $f^{(\pi)}(\hat{y}^{(\pi)}) - f^{(\pi)}(\bar{y}^{(\pi)})$ is sufficiently large compared to the predicted decrease

$$\begin{aligned} \Delta^{(\pi)}(\bar{w}^{(\pi)}, \hat{y}^{(\pi)}) :=& f^{(\pi)}(\hat{y}^{(\pi)}) - \hat{f}^{(\pi)}_{\bar{w}^{(\pi)}}(\bar{y}^{(\pi)}) \\ =& f^{(\pi)}(\hat{y}^{(\pi)}) - \hat{f}^{(\pi)}_{\bar{w}^{(\pi)}}(\hat{y}^{(\pi)}) + \frac{1}{u} \|g^{(\pi)}(\bar{x}^{(\pi)})\|^2 \\ =& \sum_{r \in R_{J^{(\pi)}}} \left[f_r^{(\pi)}(\hat{y}^{(\pi)}) - \hat{f}^{(\pi)}_{\bar{w}_r^{(\pi)},r}(\hat{y}^{(\pi)}) \right] + \frac{1}{u} \|g^{(\pi)}(\bar{x}^{(\pi)})\|^2. \end{aligned} \tag{5.42}$$

The subspace progress $\Delta^{(\pi)}(\bar{w}^{(\pi)}, \hat{y}^{(\pi)})$ has a direct relation to the expected progress $\Delta(\bar{w}, \hat{y})$ of the global problem, in particular to its part $\Delta_{J^{(\pi)}}(\bar{w}, \hat{y})$.

Observation 5.16 *Let π be a subspace problem with subspace $J^{(\pi)} \subseteq M$. Then for all $\bar{w} \in \operatorname{conv} W$ it is*

$$\Delta_{J^{(\pi)}}(\bar{w}, \hat{y}^{\langle\pi\rangle}) = \Delta^{(\pi)}(\bar{w}_{R_{J^{(\pi)}}}, \hat{y}_J^{\langle\pi\rangle}).$$

In other words, setting $\bar{w} = \bar{w}^{\langle\pi\rangle}$, the subspace problem is setup so that its expected progress is exactly the progress that the global problem expects on the subspace $J^{(\pi)}$.

Proof. Using the trivial fact $\hat{y}_{\bar{J}}^{\langle\pi\rangle} = \hat{y}_{\bar{J}}^{\langle\pi\rangle}$ we can apply Observation 5.15 and by definition (5.42) of $\Delta^{(\pi)}(\bar{w}_{R_{J^{(\pi)}}}, y^{(\pi)})$ we get with $\bar{w} = (\bar{l}, \bar{x})$

$$\begin{aligned}
\Delta^{(\pi)}(\bar{w}_{R_{J^{(\pi)}}}, \hat{y}_{J^{(\pi)}}^{\langle\pi\rangle}) &= \sum_{r \in R_{J^{(\pi)}}} \left[f_r^{(\pi)}(\hat{y}_{J^{(\pi)}}^{\langle\pi\rangle}) - \hat{f}_{\bar{w}_r, r}^{(\pi)}(\hat{y}_{J^{(\pi)}}^{\langle\pi\rangle}) \right] + \frac{1}{u} \|g^{(\pi)}(\bar{x}_{R_{J^{(\pi)}}})\|^2 \\
&= \sum_{r \in R_{J^{(\pi)}}} \left[f_r(\hat{y}^{\langle\pi\rangle}) - \hat{f}_{\bar{w}_r, r}(\hat{y}^{\langle\pi\rangle}) \right] + \frac{1}{u} \|g(\bar{x})_{J^{(\pi)}}\|^2 \\
&\overset{(5.27)}{=} \Delta_{J^{(\pi)}}(\bar{w}, \hat{y}^{\langle\pi\rangle}).
\end{aligned}$$

□

We will not solve the subspace problem to optimality (or close to the optimal value). Instead it is only solved until its solution promises a sufficient improvement for the global problem. In particular, given a parameter $\tau_2 \in (0, 1)$, we stop if either the predicted decrease has been sufficiently reduced, *i. e.*

$$\Delta^{(\pi)}(\bar{w}^{(\pi)}, \hat{y}_{J^{(\pi)}}^{\langle\pi\rangle}) < \tau_2 \Delta_{J^{(\pi)}}(\bar{w}^{\langle\pi\rangle}, \hat{y}^{\langle\pi\rangle}), \tag{X1}$$

or a descent step occurs, *i. e.*

$$\Delta^{(\pi)}(\bar{w}^{(\pi)}, \hat{y}_{J^{(\pi)}}^{\langle\pi\rangle}) \leq \frac{1}{\varrho}(f^{(\pi)}(\hat{y}_{J^{(\pi)}}^{\langle\pi\rangle}) - f^{(\pi)}(\bar{y}^{(\pi)})), \tag{X2}$$

where $\bar{y}^{(\pi)} \in \mathbb{R}^{J^{(\pi)}}$ is the current candidate of the bundle algorithm. Note that these conditions will be checked in order, so that

$$\Delta^{(\pi)}(\bar{w}^{(\pi)}, \hat{y}_{J^{(\pi)}}^{\langle\pi\rangle}) \geq \tau_2 \Delta_{J^{(\pi)}}(\bar{w}^{\langle\pi\rangle}, \hat{y}^{\langle\pi\rangle})$$

if a descent step occurs.

The algorithm solving a subspace problem is shown in Algorithm 5.3. A first observation ensures that no subprocess runs forever.

Observation 5.17 *Algorithm 5.3 is finite.*

Algorithm 5.3: SolveSubspace

Input : process data π
Output : **TRUE** if a descent step occurs
Changes: set final values $\bar{w}^{(\pi)}$, $\hat{y}^{(\pi)}$, $f^{(\pi)}_{R_{J^{(\pi)}}}$

Starting from initial centre $\hat{y}^{(\pi)} = \hat{y}^{\langle\pi\rangle}_{J^{(\pi)}}$ and initial model $\widehat{W}^{(\pi)} = \{\bar{w}^{\langle\pi\rangle}_{R_{J^{(\pi)}}}\}$,
solve (Sub(π)) until either (X1) or (X2) is fulfilled.
This gives final

- aggregated minorant $\bar{w}^{(\pi)} \in \operatorname{conv} W^{(\pi)}$,
- final centre $\hat{y}^{(\pi)} \in \mathbb{R}^{J^{(\pi)}}$,
- function values $f^{(\pi)}_{R_{J^{(\pi)}}}(\hat{y}^{(\pi)})$.

if *(X1) is fulfilled* **then**
  // $\hat{y}^{(\pi)} = \hat{y}^{\langle\pi\rangle}_{J^{(\pi)}}$ is the old centre, $f^{(\pi)}_{R_{J^{(\pi)}}}(\hat{y}^{(\pi)}) = f^{\langle\pi\rangle}_{R_{J^{(\pi)}}}$ are the values in the old centre
  return *FALSE*
else
  // (X2) is fulfilled
  // $\hat{y}^{(\pi)}$ is the new centre, $f^{(\pi)}_{R_{J^{(\pi)}}}(\hat{y}^{(\pi)})$ the values in the new centre
  return *TRUE*

Proof. The algorithm is a standard proximal bundle algorithm, thus by Theorem 5.3 either a descent step occurs or the predicted progress $\Delta^{(\pi)}$ tends to zero. Consequently either (X1) or (X2) will be satisfied in finite time. □

The following simple but important observation shows a consequence of the two stopping conditions. Whenever condition (X2) stops the algorithm then the decrease in $f^{(\pi)}(y^{(\pi)})$ is large compared with the progress $\Delta_{J^{(\pi)}}(\bar{w}^{\langle\pi\rangle}, \hat{y}^{\langle\pi\rangle})$, which has been promised on the subspace $J^{(\pi)}$ when the subspace has been selected.

Observation 5.18 *Let π be a subspace problem with subspace $J \subseteq M$, $\bar{w}^{\langle\pi\rangle} \in \operatorname{conv} W$ and $\hat{y}^{\langle\pi\rangle} \in \mathbb{R}^M$, and assume that Algorithm 5.3 stopped because of (X2) with final values $\bar{w}^{(\pi)} \in \operatorname{conv} W^{(\pi)}$ and $\bar{y}^{(\pi)} \in \mathbb{R}^{J^{(\pi)}}$. Then*

$$f^{(\pi)}(\hat{y}^{\langle\pi\rangle}_{J^{(\pi)}}) - f^{(\pi)}(\bar{y}^{(\pi)}) \geq \tau_2 \varrho \Delta_{J^{(\pi)}}(\bar{w}^{\langle\pi\rangle}, \hat{y}^{\langle\pi\rangle}).$$

Proof. Since the algorithm stopped because of (X2) and *not* because of (X1) we know

$$\Delta^{(\pi)}(\bar{w}^{(\pi)}, \hat{y}^{\langle\pi\rangle}_{J^{(\pi)}}) \geq \tau_2 \Delta_{J^{(\pi)}}(\bar{w}^{\langle\pi\rangle}, \hat{y}^{\langle\pi\rangle})$$

and

$$\Delta^{(\pi)}(\bar{w}^{(\pi)}, \hat{y}^{\langle\underline{\pi}\rangle}_{J^{(\pi)}}) \leq \frac{1}{\varrho}(f^{(\pi)}(\hat{y}^{\langle\underline{\pi}\rangle}_{J^{(\pi)}}) - f^{(\pi)}(\bar{y}^{(\pi)})).$$

The result follows by putting these two inequalities together. □

5.3.6 Update of the Global Data

The last step of a subprocess solving some subspace problem π is the update of the global data. Depending on the outcome of Algorithm 5.3, *i. e.* whether a descent step occurred or not, the update is slightly different. If the algorithm stopped without a descent step then the predicted progress on the subspace has not been good. In this case an additional step detects potential dependencies between the subspace and constraints that do not belong to the subspace. In particular, we test if the following condition holds for $\tau_3 \in (0, 1-\tau_2)$

(Dep) $$\delta_{\bar{J}^{(\pi)}}(\bar{w}^{\langle\bar{\pi}+1\rangle}) - \delta_{\bar{J}^{(\pi)}}(\bar{w}^{\langle\bar{\pi}\rangle}) > \tau_3 \Delta_{J^{(\pi)}}(\bar{w}^{\langle\underline{\pi}\rangle}, \hat{y}^{\langle\underline{\pi}\rangle}).$$

The motivation of this condition is as follows. By Observation 5.8 the expected progress can be split intro three terms w. r. t. a subspace $J^{(\pi)} \subseteq M$

$$\Delta(\bar{w}^{\langle\sigma\rangle}, \hat{y}^{\langle\sigma\rangle}) = \Delta_{J^{(\pi)}}(\bar{w}^{\langle\sigma\rangle}, \hat{y}^{\langle\sigma\rangle}) + \bar{\Delta}_{J^{(\pi)}}(\bar{w}^{\langle\sigma\rangle}, \hat{y}^{\langle\sigma\rangle}) + \delta_{\bar{J}^{(\pi)}}$$

The subprocess optimising over $J^{(\pi)}$ will decrease $\Delta_{J^{(\pi)}}(\bar{w}^{\langle\sigma\rangle}, \hat{y}^{\langle\sigma\rangle})$ and not change $\bar{\Delta}_{J^{(\pi)}}(\bar{w}^{\langle\sigma\rangle}, \hat{y}^{\langle\sigma\rangle})$. A bad dependency may increase the third term $\delta_{\bar{J}^{(\pi)}}$ by a large amount compared to the decrease of first term so that in sum the whole progress does not decrease sufficiently. Condition (Dep) detects this situation by comparing the change in the third term with the progress made in the first term. The details and a formal verification of these intuitive statements are given in Sections 5.3.8 and 5.3.9. If a dependency is detected, *i. e.*, if (Dep) does not hold, the dependency graph D is increased by an additional arc, so that this dependency is respected in future subspace selections (also see Section 5.3.4 on how the dependency graph is considered in the subspace selection step). The update is shown in Algorithm 5.4.

Remark 5.19 In A5.4.4 the new dependency arc (j, j') can always be selected, because if $J^{(\pi)} \times \bar{J}^{(\pi)} \subseteq E^{\langle\bar{\pi}\rangle}$ then additional constraints had been added to the subspace $J^{(\pi)}$ when it has been selected.

5.3.7 The Algorithm

We are now ready to put all parts together. The parallel bundle algorithm is called with the problem to be solved and the parameters τ_1, τ_2, τ_3, ε and u. In addition the
N_Π parameter N_Π is passed to the algorithm that specifies the maximal number of processes that should be started in parallel. Because each process works on a non-empty subspace and the subspaces of parallel processes are disjoint, the number of parallel processes is

Algorithm 5.4: UPDATESUBSPACE

Input : process data π,
$Descent \in \{\mathbf{TRUE}, \mathbf{FALSE}\}$
Changes: sets global data at $\bar{\pi}+1$

1 $(\bar{w}^{\langle\bar{\pi}+1\rangle}_{R_{J(\pi)}}, \bar{w}^{\langle\bar{\pi}+1\rangle}_{R\setminus R_{J(\pi)}}) \leftarrow (\bar{w}^{(\pi)}, \bar{w}^{\langle\bar{\pi}\rangle}_{R\setminus R_{J(\pi)}})$ `// Update aggregate minorant on` $R_{J(\pi)}$

2 $(\hat{y}^{\langle\bar{\pi}+1\rangle}_{J(\pi)}, \hat{y}^{\langle\bar{\pi}+1\rangle}_{M\setminus J(\pi)}) \leftarrow (\hat{y}^{(\pi)}, \hat{y}^{\langle\bar{\pi}\rangle}_{M\setminus J(\pi)})$ `// Update centre of stability on` $J^{(\pi)}$

3 $(f^{\langle\bar{\pi}+1\rangle}_{R_{J(\pi)}}, f^{\langle\bar{\pi}+1\rangle}_{R\setminus R_{J(\pi)}}) \leftarrow (f^{(\pi)}_{R_{J(\pi)}}(\hat{y}^{(\pi)}), f^{\langle\bar{\pi}\rangle}_{R\setminus R_{J(\pi)}})$ `// Update function values in centre`

if $Descent = \boldsymbol{FALSE}$ *and* $\delta_{\bar{J}(\pi)}(\bar{w}^{\langle\bar{\pi}+1\rangle}) - \delta_{\bar{J}(\pi)}(\bar{w}^{\langle\bar{\pi}\rangle}) > \tau_3 \Delta_{J(\pi)}(\bar{w}^{\langle\underline{\pi}\rangle}, \hat{y}^{\langle\underline{\pi}\rangle})$ **then**

`// Update dependency graph with` $\bar{w}^{\langle\bar{\pi}\rangle} = (\bar{l}^{\langle\bar{\pi}\rangle}, \bar{x}^{\langle\bar{\pi}\rangle})$ `and` $\bar{w}^{\langle\bar{\pi}+1\rangle} = (\bar{l}^{\langle\bar{\pi}+1\rangle}, \bar{x}^{\langle\bar{\pi}+1\rangle})$

4 choose $(j, j') \in \operatorname{Argmax}\left\{g(\bar{x}^{\langle\bar{\pi}+1\rangle})^2_{\hat{j}} - g(\bar{x}^{\langle\bar{\pi}\rangle})^2_{\hat{j}} \colon (\tilde{j}, \hat{j}) \in (J^{(\pi)} \times \bar{J}^{(\pi)}) \setminus E^{\langle\bar{\pi}\rangle}\right\}$

5 $E^{\langle\bar{\pi}+1\rangle} \leftarrow E^{\langle\bar{\pi}\rangle} \cup \{(j, j')\}$

else

6 $E^{\langle\bar{\pi}+1\rangle} \leftarrow E^{\langle\bar{\pi}\rangle}$

bounded by the number of constraints, *i. e.*, $|M|$. But in practice one would like to restrict the number of parallel processes to the number of processors available on the hardware platform to be used. The parallel bundle algorithm is shown in Algorithm 5.5.

5.3.8 Consistency of the Update Scheme

In this section we analyse the relations between the "local" variables in each subprocess and the "global" values when the subprocess starts, when it is running and when it updates the global data. We will show that the update happens in a very predictable way. The first statement verifies that the global data, that is associated with the subspace of a subprocess, remains largely unchanged during the run of this subprocess.

Lemma 5.20 *For all* $\sigma \in \Sigma$,

(i) $B^{\langle\sigma\rangle} = \bigcup_{\pi\in\Pi^{\langle\sigma\rangle}} R_{J(\pi)}$,

(ii) $R_{J(\pi)} \cap R_{J(\pi')} = \emptyset$ *for* $\pi, \pi' \in \Pi^{\langle\sigma\rangle}$ *with* $\pi \neq \pi'$,

(iii) $J^{(\pi')} \cap (J^{(\pi)} \cup \bar{J}^{(\pi)}) = J^{(\pi)} \cap (J^{(\pi')} \cup \bar{J}^{(\pi')}) = \emptyset$ *for* $\pi, \pi' \in \Pi^{\langle\sigma\rangle}$ *with* $\pi \neq \pi'$,

(iv) $\hat{y}^{\langle\underline{\pi}\rangle}_{J(\pi)\cup\bar{J}(\pi)} = \hat{y}^{\langle\sigma\rangle}_{J(\pi)\cup\bar{J}(\pi)}$, $\bar{w}^{\langle\underline{\pi}\rangle}_{R_{J(\pi)}} = \bar{w}^{\langle\sigma\rangle}_{R_{J(\pi)}}$, *and* $f^{\langle\underline{\pi}\rangle}_{R_{J(\pi)}} = f^{\langle\sigma\rangle}_{R_{J(\pi)}}$ *for all* $\pi \in \Pi^{\langle\sigma\rangle}$.

Algorithm 5.5: ParallelBundle

Input : Problem (P), parameters
- $\tau_1 \in (0, \frac{1}{2}\min\{\frac{1}{|M|}, \frac{1}{|R|}\})$, $\tau_2 \in (0,1)$, $\tau_3 \in [0, 1-\tau_2)$, $\varepsilon > 0$, $u > 0$
- $N_\Pi \in \{1, \ldots, |M|\}$

Output: Approximate solutions $x \in \operatorname{conv} \mathcal{X}$, $y \in \mathbb{R}^M$

`// Initialisation`

set $\sigma \leftarrow 0$, $\hat{y}^{\langle 0\rangle} \leftarrow 0$, $B^{\langle 0\rangle} \leftarrow \emptyset$

set $E^{\langle 0\rangle} \leftarrow \emptyset$ (or use some pre-specified dependencies)

set $f_r^{\langle 0\rangle} \leftarrow f_r(\hat{y}^{\langle 0\rangle})$, $r \in R$

set $\bar{w}^{\langle 0\rangle} = (\bar{l}^{\langle 0\rangle}, \bar{x}^{\langle 0\rangle})$ to the minorant defined by the optimal solutions

set $\Delta^{\langle 0\rangle} \leftarrow \Delta(\bar{w}^{\langle 0\rangle}, \hat{y}^{\langle 0\rangle})$

1 **if** $\Delta^{\langle 0\rangle} \le \varepsilon(|f(\hat{y}^{\langle 0\rangle})| + 1)$ **then**

  then do not start any process

  set $x \leftarrow \bar{x}^{\langle 0\rangle}$, $y \leftarrow \hat{y}^{\langle 0\rangle}$

  return

while *Less than* N_Π *processes are running* **do**

  Start a new process $\pi = (\underline{\pi}, \bar{\pi}) \leftarrow (-1, -1)$.

  Each *process* π *performs the following steps independently*

    Secure exclusive access to global data

      $\underline{\pi} \leftarrow \sigma$

      if SelectSubspace*(*$\underline{\pi}$*)* = ***FALSE*** **then**

        `// Step is unsuccessful`

        Free exclusive access to global data

        STOP this process

2 $(\bar{w}^{\langle \underline{\pi}+1\rangle}, \hat{y}^{\langle \underline{\pi}+1\rangle}, f^{\langle \underline{\pi}+1\rangle}, E^{\langle \underline{\pi}+1\rangle}) \leftarrow (\bar{w}^{\langle \underline{\pi}\rangle}, \hat{y}^{\langle \underline{\pi}\rangle}, f^{\langle \underline{\pi}\rangle}, E^{\langle \underline{\pi}\rangle})$

3 $B^{\langle \underline{\pi}+1\rangle} \leftarrow B^{\langle \underline{\pi}\rangle} \cup R_{J^{(\pi)}}$

  $\bar{\pi} \leftarrow \infty$, $\sigma \leftarrow \underline{\pi} + 1$

    Free exclusive access to global data

    `// Solve the subspace problem to sufficient precision`

4 *Descent* $\leftarrow$ SolveSubspace(π)

    Secure exclusive access to global data

      $\bar{\pi} \leftarrow \sigma$

      UpdateSubspace$(\pi,$ *Descent*$)$

5 $B^{\langle \bar{\pi}+1\rangle} \leftarrow B^{\langle \bar{\pi}\rangle} \setminus R_{J^{(\pi)}}$

      $\sigma \leftarrow \bar{\pi} + 1$

6 **if** $\Delta(\bar{w}^{\langle \bar{\pi}+1\rangle}, \hat{y}^{\langle \bar{\pi}+1\rangle}) \le \varepsilon(|f(\hat{y}^{\langle \bar{\pi}+1\rangle})| + 1)$ **then**

        set $x \leftarrow \bar{x}^{\langle \sigma\rangle}$, $y \leftarrow \hat{y}^{\langle \sigma\rangle}$

        TERMINATE *all* processes and **STOP**.

    Free exclusive access to global data

    STOP this process π.

Proof. The proof works by induction on σ. For $\sigma = 0$ we have $B^{\langle\sigma\rangle} = \emptyset$ and $\Pi^{\langle\sigma\rangle} = \emptyset$, thus (i)–(iv) hold trivially. Now suppose $\sigma + 1 \in \Sigma$ and the claim holds for $\sigma \in \Sigma$. By definition, $\sigma + 1 \in \Sigma$ implies $\Pi^{\langle\sigma\rangle} \neq \Pi^{\langle\sigma+1\rangle}$, so Observation 5.10 asserts the existence of a unique $\eta \in (\Pi^{\langle\sigma\rangle} \setminus \Pi^{\langle\sigma+1\rangle}) \cup (\Pi^{\langle\sigma+1\rangle} \setminus \Pi^{\langle\sigma\rangle})$ and this η either satisfies $\underline{\eta} = \sigma$ or $\bar{\eta} = \sigma$.

If $\underline{\eta} = \sigma$ we have $\Pi^{\langle\sigma\rangle} = \Pi^{\langle\sigma+1\rangle} \setminus \{\eta\}$ and process η executed a successful subspace selection step (Algorithm 5.2) at σ. Thus the process executes lines A5.5.2 and A5.5.3. This implies $B^{\langle\sigma+1\rangle} = B^{\langle\sigma\rangle} \cup R_{J(\eta)}$ and consequently (i) as well as (iv) hold. By induction, (ii) only needs to be verified for $\pi = \eta$ and $\pi' \in \Pi^{\langle\sigma\rangle}$. On the one hand (i) implies $R_{J(\pi')} \subseteq B^{\langle\sigma\rangle}$ and on the other hand $R_{J(\eta)} \cap B^{\langle\sigma\rangle} = \emptyset$ by Observation 5.14, hence (ii) holds. Finally (iii) follows from (ii) and Observation 5.11.

If $\bar{\eta} = \sigma$ we have $\Pi^{\langle\sigma+1\rangle} = \Pi^{\langle\sigma\rangle} \setminus \{\eta\}$ and process η called Algorithm 5.4 and executed line A5.5.5 at σ. The latter implies $B^{\langle\sigma+1\rangle} = B^{\langle\sigma\rangle} \setminus R_{J(\eta)}$ and with (ii) at σ relation (i) follows. In view of the validity of (ii) and (iii) at σ, both hold by induction also for $\sigma + 1$ and its remaining processes. The new values for $\bar{w}^{\langle\sigma+1\rangle}$, $\hat{y}^{\langle\sigma+1\rangle}$ and $f^{\langle\sigma+1\rangle}$ are set in lines A5.4.1 – A5.4.3. In particular, the values for indexes $R \setminus R_{J(\eta)}$ resp. $M \setminus J^{(\eta)}$ are unchanged. Together with (ii) and (iii) for σ this proves (iv). □

Next we assert that, when a process π stops at $\bar{\pi}$, the relevant subspace information for π has not been modified within the global data throughout its runtime and at $\bar{\pi} + 1$ the data on its selected subspace and subproblems is consistent with the terminal status of the bundle method of π. In the following we use the short notation for each $\sigma \in \mathbb{N}_0$ and each subspace $J \subseteq M$.

$$\Delta^{\langle\sigma\rangle} := \Delta(\bar{w}^{\langle\sigma\rangle}, \hat{y}^{\langle\sigma\rangle}), \qquad \Delta^{\langle\sigma\rangle}$$

$$\Delta_J^{\langle\sigma\rangle} := \Delta_J(\bar{w}^{\langle\sigma\rangle}, \hat{y}^{\langle\sigma\rangle}), \qquad \Delta_J^{\langle\sigma\rangle}$$

$$\bar{\Delta}_J^{\langle\sigma\rangle} := \bar{\Delta}_J(\bar{w}^{\langle\sigma\rangle}, \hat{y}^{\langle\sigma\rangle}), \qquad \bar{\Delta}_J^{\langle\sigma\rangle}$$

$$\delta_{\bar{J}}^{\langle\sigma\rangle} := \delta_{\bar{J}}(\bar{w}^{\langle\sigma\rangle}). \qquad \delta_{\bar{J}}^{\langle\sigma\rangle}$$

Lemma 5.21 *Given $\pi \in \bar{\Pi}^{\langle\infty\rangle}$ assume $f_{R_{J(\pi)}}^{\langle\underline{\pi}\rangle} = f_{R_{J(\pi)}}(\hat{y}^{\langle\underline{\pi}\rangle})$. Then*

$$\hat{y}_{\bar{J}(\pi)}^{\langle\underline{\pi}\rangle} = \hat{y}_{\bar{J}(\pi)}^{\langle\sigma\rangle} \qquad \text{for all } \sigma \in \{\underline{\pi}, \dots, \bar{\pi} + 1\}, \tag{5.43}$$

$$\hat{y}_{J(\pi)}^{\langle\underline{\pi}\rangle} = \hat{y}_{J(\pi)}^{\langle\sigma\rangle} \qquad \text{for all } \sigma \in \{\underline{\pi}, \dots, \bar{\pi}\}, \tag{5.44}$$

$$f_{R_{J(\pi)}}^{\langle\underline{\pi}\rangle} = f_{R_{J(\pi)}}^{\langle\sigma\rangle} = f_{R_{J(\pi)}}(\hat{y}^{\langle\sigma\rangle}) = f_{R_{J(\pi)}}^{(\pi)}(\hat{y}_{J(\pi)}^{\langle\sigma\rangle}) \qquad \text{for all } \sigma \in \{\underline{\pi}, \dots, \bar{\pi}\}, \tag{5.45}$$

$$\bar{w}_{R_{J(\pi)}}^{\langle\underline{\pi}\rangle} = \bar{w}_{R_{J(\pi)}}^{\langle\sigma\rangle} \qquad \text{for all } \sigma \in \{\underline{\pi}, \dots, \bar{\pi}\}, \tag{5.46}$$

$$\Delta_{J(\pi)}^{\langle\underline{\pi}\rangle} = \Delta_{J(\pi)}^{\langle\sigma\rangle} \qquad \text{for all } \sigma \in \{\underline{\pi}, \dots, \bar{\pi}\}, \tag{5.47}$$

and with $\bar{w}^{(\pi)} \in \operatorname{conv} W^{(\pi)}$, $\hat{y}^{(\pi)} \in \mathbb{R}^{J^{(\pi)}}$ *and* $f^{(\pi)}_{R_{J^{(\pi)}}}(\hat{y}^{(\pi)}) \in \mathbb{R}^{R_{J^{(\pi)}}}$ *the values of* π *returned by the call to subroutine Algorithm 5.3*

$$\hat{y}^{\langle\bar{\pi}+1\rangle}_{J^{(\pi)}} = \hat{y}^{(\pi)}, \qquad f^{\langle\bar{\pi}+1\rangle}_{R_{J^{(\pi)}}} = f^{(\pi)}_{R_{J^{(\pi)}}}(\hat{y}^{(\pi)}) = f_{R_{J^{(\pi)}}}(\hat{y}^{\langle\bar{\pi}+1\rangle}), \tag{5.48}$$

$$\hat{y}^{\langle\bar{\pi}+1\rangle}_{M\setminus J^{(\pi)}} = \hat{y}^{\langle\bar{\pi}\rangle}_{M\setminus J^{(\pi)}}, \qquad f^{\langle\bar{\pi}+1\rangle}_{R\setminus R_{J^{(\pi)}}} = f^{\langle\bar{\pi}\rangle}_{R\setminus R_{J^{(\pi)}}}, \tag{5.49}$$

$$\bar{w}^{\langle\bar{\pi}+1\rangle}_{R_{J^{(\pi)}}} = \bar{w}^{(\pi)}, \qquad \bar{w}^{\langle\bar{\pi}+1\rangle}_{R\setminus R_{J^{(\pi)}}} = \bar{w}^{\langle\bar{\pi}\rangle}_{R\setminus R_{J^{(\pi)}}}, \tag{5.50}$$

$$\Delta^{\langle\bar{\pi}+1\rangle}_{J^{(\pi)}} = \Delta^{(\pi)}(\bar{w}^{(\pi)}, \hat{y}^{(\pi)}), \qquad \bar{\Delta}^{\langle\bar{\pi}+1\rangle}_{J^{(\pi)}} = \bar{\Delta}^{\langle\bar{\pi}\rangle}_{J^{(\pi)}}, \tag{5.51}$$

Proof. For $\sigma \in \{\underline{\pi}+1, \dots, \bar{\pi}\}$ we have $\pi \in \Pi^{\langle\sigma\rangle}$, hence, for these values of σ, Lemma 5.20 (iv) implies (5.43), (5.44), (5.46), and the first equation of (5.45) (the remaining two will be proved below). With these (5.47) follows from the definition (5.27) together with (5.23)–(5.25) using Observation 5.6.

Process π updates the global data for index $\sigma = \bar{\pi} + 1$ in lines A5.4.1 – A5.4.3. The values for the indexes $R_{J^{(\pi)}}$ resp. $J^{(\pi)}$ are set to the final values computed by Algorithm 5.3, *i. e.*, $\bar{w}^{(\pi)}$, $\hat{y}^{(\pi)}$ and $f^{(\pi)}_{R_{J^{(\pi)}}}$, whereas the values for the other indexes remain unchanged. This proves (5.49) and (5.50) and the first two equations of (5.48) and completes the proof for (5.43) because $\bar{J}^{(\pi)} \subseteq M \setminus J^{(\pi)}$.

Note that for the rest of the proof we may invoke Observation 5.15 for $\hat{y}^{\langle\sigma\rangle}$ and $\sigma \in \{\underline{\pi}, \dots, \bar{\pi}+1\}$ because of (5.43). In particular, to complete (5.45) and (5.48), observe for any $\sigma \in \{\underline{\pi}, \dots, \bar{\pi}\}$

$$f_{R_{J^{(\pi)}}}(\hat{y}^{\langle\sigma\rangle}) \overset{(5.39)}{=} f^{(\pi)}_{R_{J^{(\pi)}}}(\hat{y}^{\langle\sigma\rangle}_{J^{(\pi)}}) \overset{(5.44)}{=} f^{(\pi)}_{R_{J^{(\pi)}}}(\hat{y}^{\langle\underline{\pi}\rangle}_{J^{(\pi)}}) \overset{(5.39)}{=} f_{R_{J^{(\pi)}}}(\hat{y}^{\langle\underline{\pi}\rangle}) = f^{\langle\underline{\pi}\rangle}_{R_{J^{(\pi)}}}$$

(the last equation holds by assumption), and

$$f_{R_{J^{(\pi)}}}(\hat{y}^{\langle\bar{\pi}+1\rangle}) \overset{(5.39)}{=} f^{(\pi)}_{R_{J^{(\pi)}}}(\hat{y}^{\langle\bar{\pi}+1\rangle}_{J^{(\pi)}}).$$

It remains to show (5.51). The left-hand side equation of (5.51) follows by

$$\begin{aligned}
\Delta^{(\pi)}(\bar{w}^{(\pi)}, \hat{y}^{(\pi)}) &\overset{(5.42)}{=} \sum_{r\in R_{J^{(\pi)}}} \Big[f^{(\pi)}_r(\hat{y}^{(\pi)}) - \hat{f}^{(\pi)}_{\bar{w}^{(\pi)},r}(\hat{y}^{(\pi)})\Big] + \frac{1}{u}\|g^{(\pi)}(\bar{x}^{(\pi)})\|^2 \\
&\overset{(5.48),(5.50)}{=} \sum_{r\in R_{J^{(\pi)}}} \Big[f^{\langle\bar{\pi}+1\rangle}_r - \hat{f}^{(\pi)}_{\bar{w}^{\langle\bar{\pi}+1\rangle}_r,r}(\hat{y}^{\langle\bar{\pi}+1\rangle}_{J^{(\pi)}})\Big] + \frac{1}{u}\|g^{(\pi)}(\bar{x}^{\langle\bar{\pi}+1\rangle}_{R_{J^{(\pi)}}})\|^2 \\
&\overset{(5.40),(5.41)}{=} \sum_{r\in R_{J^{(\pi)}}} \Big[f^{\langle\bar{\pi}+1\rangle}_r - \hat{f}_{\bar{w}^{\langle\bar{\pi}+1\rangle}_r,r}(\hat{y}^{\langle\bar{\pi}+1\rangle})\Big] + \frac{1}{u}\|g(\bar{x}^{\langle\bar{\pi}+1\rangle})_{J^{(\pi)}}\|^2 \\
&= \Delta^{\langle\bar{\pi}+1\rangle}_{J^{(\pi)}}.
\end{aligned}$$

In order to show $\bar{\Delta}_{J^{(\pi)}}^{\langle\bar{\pi}+1\rangle} = \bar{\Delta}_{J^{(\pi)}}^{\langle\bar{\pi}\rangle}$ it suffices to check that none of the values involved in (5.28) change when π executes Algorithm 5.4 at $\bar{\pi}$. This follows from (5.49) and (5.50) because for $r \in R \setminus R_{J^{(\pi)}}$ we have $J_r \subseteq M \setminus J^{(\pi)}$ by Observation 5.6 and so $\hat{y}_{J_r}^{\langle\bar{\pi}+1\rangle} = \hat{y}_{J_r}^{\langle\bar{\pi}\rangle}$, $f_r^{\langle\bar{\pi}+1\rangle} = f_r^{\langle\bar{\pi}\rangle}$, and $\bar{w}_r^{\langle\bar{\pi}+1\rangle} = \bar{w}_r^{\langle\bar{\pi}\rangle}$. Thus, (5.24) establishes $\hat{f}_{\bar{w}_r^{\langle\bar{\pi}+1\rangle},r}(\hat{y}^{\langle\bar{\pi}+1\rangle}) = \hat{f}_{\bar{w}_r^{\langle\bar{\pi}\rangle},r}(\hat{y}^{\langle\bar{\pi}\rangle})$ for $r \in R \setminus R_{J^{(\pi)}}$, while $g(\bar{x}^{\langle\bar{\pi}+1\rangle})_{M\setminus(J^{(\pi)}\cup\bar{J}^{(\pi)})} = g(\bar{x}^{\langle\bar{\pi}\rangle})_{M\setminus(J^{(\pi)}\cup\bar{J}^{(\pi)})}$ follows via (5.25) because $R_{M\setminus(J^{(\pi)}\cup\bar{J}^{(\pi)})} \cap R_{J^{(\pi)}} = \emptyset$ (by Observation 5.6). Thus, $\bar{\Delta}_{J^{(\pi)}}^{\langle\bar{\pi}+1\rangle} = \bar{\Delta}_{J^{(\pi)}}^{\langle\bar{\pi}\rangle}$ and (5.51) holds. □

Throughout the algorithm, the global data is consistent and the arc set of the dependency graph may only increase.

Lemma 5.22 *For all $\sigma \in \Sigma$ it holds,*

(i) $f_R^{\langle\sigma\rangle} = f_R(\hat{y}^{\langle\sigma\rangle})$,

(ii) $\bar{w}^{\langle\sigma\rangle} \in \operatorname{conv} W$,

(iii) $E^{\langle\sigma\rangle} \subseteq E^{\langle\sigma+1\rangle} \subseteq \{(i,j)\colon i,j \in M, i \neq j\}$,

Proof. The proof is by induction on σ. For $\sigma = 0$ the claim holds by the initialisation step. So suppose now $\sigma + 1 \in \Sigma$ and the claim holds for $\sigma \in \Sigma$. If $\sigma + 1$ is reached by a subspace selection step, *i. e.* $\sigma = \underline{\pi}$ for some process π then none of the involved variables are changed because of line A5.5.2 and all relations still hold. Otherwise $\sigma + 1$ is reached by an update step executed by some process π with $\bar{\pi} = \sigma$. By induction hypothesis and $\underline{\pi} < \bar{\pi} = \sigma$ we can apply Lemma 5.21. For $r \in R \setminus R_{J^{(\pi)}}$ Observation 5.6 yields $J_r \subseteq M \setminus J^{(\pi)}$ and Lemma 5.21 implies

$$\hat{y}_{J_r}^{\langle\bar{\pi}+1\rangle} = \hat{y}_{J_r}^{\langle\bar{\pi}\rangle}, \qquad f_r^{\langle\bar{\pi}+1\rangle} = f_r^{\langle\bar{\pi}\rangle}, \qquad \bar{w}_r^{\langle\bar{\pi}+1\rangle} = \bar{w}_r^{\langle\bar{\pi}\rangle}.$$

So for $r \in R \setminus R_{J^{(\pi)}}$ (i) and (ii) hold by induction and (5.23). For $r \in R_{J^{(\pi)}}$ claims (i) and (ii) follow directly from (5.48) and (5.50) because $\bar{w}^{(\pi)} \in \operatorname{conv} W^{(\pi)} = \operatorname{conv} W_{R_{J^{(\pi)}}}$.

Relation (iii) follows directly from lines A5.5.2, A5.4.5 and A5.4.6. □

Next we show that the algorithm terminates if and only if the global data satisfies the termination criterion. This is not completely obvious because the algorithm consists of several parallel processes. Without care it could happen that the algorithm does not terminate but the global data does not change anymore and no new processes are started successfully. This might be the case if no valid subspace can be selected anymore so that all subspace selection steps fail or if the algorithm is in some dead-lock situation. The following result verifies that this cannot happen.

Observation 5.23 *For each $\sigma \in \Sigma$ the following statements are equivalent:*

(i) $\sigma = \max \Sigma$,

(ii) $\Pi^{\langle\sigma\rangle} = \Pi^{\langle\sigma+1\rangle}$,

(iii) the algorithm terminated *with σ being the last global index,*

(iv) $\Delta(\bar{w}^{\langle\sigma\rangle}, \hat{y}^{\langle\sigma\rangle}) \leq \varepsilon(|f(\hat{y}^{\langle\sigma\rangle})|+1)$ *and* $\Delta(\bar{w}^{\langle\sigma'\rangle}, \hat{y}^{\langle\sigma'\rangle}) > \varepsilon(|f(\hat{y}^{\langle\sigma'\rangle})|+1)$ *for all* $\sigma' < \sigma$.

In particular, if $\max \Sigma = \infty$ *then the algorithm does not terminate and* $\Delta(\bar{w}^{\langle\sigma\rangle}, \hat{y}^{\langle\sigma\rangle}) > \varepsilon(|f(\hat{y}^{\langle\sigma\rangle})| + 1)$ *for all* $\sigma \in \mathbb{N}_0 = \Sigma$ *and vice versa.*

Proof. We prove this by induction on σ assuming that the equivalence holds for smaller global indexes.

(i) $\iff$ (ii): This follows by definition of Σ and Observation 5.10.

(iii) $\Rightarrow$ (iv): If $\sigma = 0$ then the algorithm must have been terminated in line A5.5.1, thus (iv) follows. If $\sigma > 0$ then by induction $\Delta(\bar{w}^{\langle\sigma'\rangle}, \hat{y}^{\langle\sigma'\rangle}) > \varepsilon(|f(\hat{y}^{\langle\sigma'\rangle})|+1)$ (otherwise σ' has been the last global index). The only possibility to terminate the algorithm is that the test in line A5.5.6 succeeds for some process π with $\bar{\pi} = \sigma - 1$.

(iv) $\Rightarrow$ (iii): (iv) implies that either $\sigma = 0$ or $\sigma > 0$ and $(\bar{w}^{\langle\sigma-1\rangle}, \hat{y}^{\langle\sigma-1\rangle}) \neq (\bar{w}^{\langle\sigma\rangle}, \hat{y}^{\langle\sigma\rangle})$. In the first case the termination test in A5.5.1 succeeds and the algorithm terminates in the initialisation step. In the second case there must be a process π with $\bar{\pi} = \sigma - 1$ setting the values for $\sigma = \bar{\pi} + 1$ in lines A5.4.1 – A5.4.3 (because if $\underline{\pi} = \sigma - 1$ the process would execute line A5.5.2 and $(\bar{w}^{\langle\sigma-1\rangle}, \hat{y}^{\langle\sigma-1\rangle}) = (\bar{w}^{\langle\sigma\rangle}, \hat{y}^{\langle\sigma\rangle})$). Consequently the termination test in line A5.5.6 will succeed and the algorithm will be terminated.

(iii) $\Rightarrow$ (ii): If the algorithm terminates with last global index σ then no process will ever execute a subspace selection step or an update step at some later index $\sigma' \geq \sigma$, hence $\Pi^{\langle\sigma\rangle} = \Pi^{\langle\sigma+1\rangle}$ by Observation 5.10.

(ii) $\Rightarrow$ (iii): (ii) implies that no process ever executes a *successful* subspace selection step or an update step at some index $\sigma' \geq \sigma$. Assume that the algorithm has not been terminated with last global index σ. We consider two cases. First if $\exists\, \pi \in \Pi^{\langle\sigma\rangle} \neq \emptyset$ then at least one process is still running at σ. Because each subspace optimisation is finite by Observation 5.17 this process will eventually execute its update step at $\bar{\pi} \geq \sigma$ and would increase σ, a contradiction. Now assume $\Pi^{\langle\sigma\rangle} = \emptyset$. In particular this is the case if $\sigma = 0$. Then Lemma 5.20 implies $B^{\langle\sigma\rangle} = \emptyset$. The algorithm will therefore try to start a new process π at $\underline{\pi} = \sigma$ and because of Observation 5.14 this step will be successful. This causes again σ to be increased, a contradiction. □

5.3.9 Convergence Analysis

Lemma 5.24 *For* $\pi \in \bar{\Pi}^{\langle\infty\rangle}$

$$0 \leq f^{(\pi)}(\hat{y}^{\langle\bar{\pi}\rangle}_{J(\pi)}) - f^{(\pi)}(\hat{y}^{\langle\bar{\pi}+1\rangle}_{J(\pi)}) = f(\hat{y}^{\langle\bar{\pi}\rangle}) - f(\hat{y}^{\langle\bar{\pi}+1\rangle}),$$

i. e., the global progress achieved when π *stores its subspace solution in the global data is exactly the progress made by* π *on its subspace problem. In particular, the sequence* $(f(\hat{y}^{\langle\sigma\rangle}))_\sigma$ *is non-increasing.*

Proof. First observe that by Lemma 5.22 the requirements for Lemma 5.21 are met, so we may use its results. We start proving the right-hand equation. It holds

$$\begin{aligned}
f^{(\pi)}(\hat{y}^{\langle\underline{\pi}\rangle}_{J(\pi)}) - f^{(\pi)}(\hat{y}^{\langle\bar{\pi}+1\rangle}_{J(\pi)}) &\overset{(5.35)}{=} b^T_{J(\pi)}(\hat{y}^{\langle\underline{\pi}\rangle}_{J(\pi)} - \hat{y}^{\langle\bar{\pi}+1\rangle}_{J(\pi)}) + \sum_{r\in R_{J(\pi)}} \left[f^{(\pi)}_r(\hat{y}^{\langle\underline{\pi}\rangle}_{J(\pi)}) - f^{(\pi)}_r(\hat{y}^{\langle\bar{\pi}+1\rangle}_{J(\pi)})\right] \\
&\overset{(5.44)}{=} b^T_{J(\pi)}(\hat{y}^{\langle\bar{\pi}\rangle}_{J(\pi)} - \hat{y}^{\langle\bar{\pi}+1\rangle}_{J(\pi)}) + \sum_{r\in R_{J(\pi)}} \left[f^{(\pi)}_r(\hat{y}^{\langle\bar{\pi}\rangle}_{J(\pi)}) - f^{(\pi)}_r(\hat{y}^{\langle\bar{\pi}+1\rangle}_{J(\pi)})\right] \\
&\overset{\text{Lemma 5.21,Lemma 5.22}}{=} b^T(\hat{y}^{\langle\bar{\pi}\rangle} - \hat{y}^{\langle\bar{\pi}+1\rangle}) + \sum_{r\in R} \left[f_r(\hat{y}^{\langle\bar{\pi}\rangle}) - f_r(\hat{y}^{\langle\bar{\pi}+1\rangle})\right] \\
&\overset{(5.17)}{=} f(\hat{y}^{\langle\bar{\pi}\rangle}) - f(\hat{y}^{\langle\bar{\pi}+1\rangle}).
\end{aligned}$$

For the left-hand inequality observe that the initial centre of the subspace problem solved by π is set to $\hat{y}^{\langle\underline{\pi}\rangle}_{J(\pi)}$ in A5.5.4 and the final centre is exactly $\hat{y}^{\langle\bar{\pi}+1\rangle}_{J(\pi)}$ by (5.48). If π stops because of condition (X1), *i. e.*, no descent step occurs, then its centre remains unchanged, *i. e.*, $\hat{y}^{\langle\bar{\pi}+1\rangle}_{J(\pi)} = \hat{y}^{(\pi)} = \hat{y}^{\langle\underline{\pi}\rangle}_{J(\pi)}$ and the inequality holds trivially. If otherwise π stops because of condition (X2), then π performs a descent step, implying the inequality as well.

Finally Observation 5.10 implies that for any $\sigma \in \Sigma$ without a $\pi' \in \bar{\Pi}^{\langle\infty\rangle}$ satisfying $\bar{\pi} = \sigma$ there is a process π'' with $\underline{\pi}'' = \sigma$ which executes a subspace selection step at $\underline{\pi}''$. Thus by A5.5.2 it is $\hat{y}^{\langle\underline{\pi}\rangle} = \hat{y}^{\langle\underline{\pi}+1\rangle}$ and consequently $f(\hat{y}^{\langle\underline{\pi}\rangle}) = f(\hat{y}^{\langle\underline{\pi}+1\rangle})$. This together with the first establishes implies that $(f(\hat{y}^{\langle\sigma\rangle}))_\sigma$ is a non-increasing sequence. □

Next we show that the algorithm always drives the predicted decrease to zero on an appropriate subsequence.

Lemma 5.25 *Suppose an infinite number of descent steps occurs and* f *is bounded from below. Then*

$$\liminf_{\sigma\in\mathbb{N}_0} \Delta^{\langle\sigma\rangle} = 0.$$

Proof. Let π be a process for which a descent step occurs, *i. e.*, π is stopped by condition (X2). The subspace selection condition (5.29) implies $\Delta^{\langle \underline{\pi} \rangle} \leq \frac{1}{\tau_1} \Delta^{\langle \underline{\pi} \rangle}_{J^{(\pi)}}$. Together with Observation 5.18 and Lemma 5.24 we get

$$\Delta^{\langle \underline{\pi} \rangle} \leq \frac{1}{\tau_1} \Delta^{\langle \underline{\pi} \rangle}_{J^{(\pi)}} \leq \frac{1}{\tau_1 \tau_2 \varrho} \left(f^{(\pi)}(\hat{y}^{\langle \underline{\pi} \rangle}_{J^{(\pi)}}) - f^{(\pi)}(\hat{y}^{\langle \bar{\pi}+1 \rangle}_{J^{(\pi)}}) \right) = \frac{1}{\tau_1 \tau_2 \varrho} \left(f(\hat{y}^{\langle \bar{\pi} \rangle}) - f(\hat{y}^{\langle \bar{\pi}+1 \rangle}) \right).$$

Because f is bounded from below and the sequence $(f(\hat{y}^{\langle \sigma \rangle}))_\sigma$ is non-increasing by Lemma 5.24, the right-hand side of the inequality above converges to zero. □

Lemma 5.26 *Assume there is only a finite number of descent steps and $\varepsilon = 0$, then*

$$\lim_{\sigma \in \Sigma} \Delta^{\langle \sigma \rangle} = 0.$$

Proof. If $|\Sigma| < \infty$, then the statement holds by Observation 5.23. Therefore we may assume $|\Sigma| = \infty$. In this case $\Pi^{\langle \sigma \rangle} \subseteq \bar{\Pi}^{\langle \infty \rangle}$ for all $\sigma \in \Sigma$ by Observation 5.17.

Observation 5.23 also implies $\Delta^{\langle \sigma \rangle} > 0$ for all $\sigma \in \Sigma$ and by Lemma 5.22 the edge set $E^{\langle \sigma \rangle}$ of the dependency graph can only be increased. Because M is a finite set there must be a $\underline{\sigma} \in \Sigma$ such that for each $\sigma \geq \underline{\sigma}$ we have $E^{\langle \sigma \rangle} = E^{\langle \underline{\sigma} \rangle}$ and all processes π with $\bar{\pi} > \underline{\sigma}$ do *not* perform a descent step. Put $\bar{\sigma} := \max\left(\{\underline{\sigma}\} \cup \{\underline{\pi}' \colon \pi' \in \Pi^{\langle \underline{\sigma} \rangle}\}\right)$. Then for π with $\bar{\pi} > \bar{\sigma}$ we have $\underline{\pi} > \underline{\sigma}$.

Let $\sigma > \underline{\sigma}$, then there is a process π such that $\sigma \in \{\underline{\pi}, \bar{\pi}\}$. If $\sigma = \underline{\pi}$ the π executes a subspace selection step and does not change $\Delta^{\langle \sigma \rangle}$ in A5.5.2, hence $\Delta^{\langle \underline{\pi} \rangle} = \Delta^{\langle \underline{\pi}+1 \rangle}$. So assume $\sigma = \bar{\pi}$. Because $\sigma > \underline{\sigma}$ process π satisfied condition (X1) and $E^{\langle \sigma+1 \rangle} = E^{\langle \sigma \rangle}$. Therefore Algorithm 5.4 did not increase the edge set and it holds

$$\delta^{\langle \bar{\pi}+1 \rangle}_{\bar{J}^{(\pi)}} - \delta^{\langle \bar{\pi} \rangle}_{\bar{J}^{(\pi)}} \leq \tau_3 \Delta^{\langle \underline{\pi} \rangle}_{J^{(\pi)}}.$$

Invoking Observation 5.8 twice for the subspace $J^{(\pi)}$ of π but once for the data of $\bar{\pi}$ and once for $\bar{\pi}+1$ yields the relations

$$\begin{aligned} \Delta^{\langle \bar{\pi} \rangle} &= \Delta^{\langle \bar{\pi} \rangle}_{J^{(\pi)}} + \delta^{\langle \bar{\pi} \rangle}_{\bar{J}^{(\pi)}} + \bar{\Delta}^{\langle \bar{\pi} \rangle}_{J^{(\pi)}}, \\ \Delta^{\langle \bar{\pi}+1 \rangle} &= \Delta^{\langle \bar{\pi}+1 \rangle}_{J^{(\pi)}} + \delta^{\langle \bar{\pi}+1 \rangle}_{\bar{J}^{(\pi)}} + \bar{\Delta}^{\langle \bar{\pi}+1 \rangle}_{J^{(\pi)}}. \end{aligned}$$

We claim that $\Delta^{\langle \bar{\pi}+1 \rangle} \leq (1-\tau)\Delta^{\langle \bar{\pi} \rangle}$ for some constant $0 < \tau < 1$ independent of π. Indeed, by Lemma 5.22 we may invoke Lemma 5.21, so (5.47) implies $\Delta^{\langle \underline{\pi} \rangle}_{J^{(\pi)}} = \Delta^{\langle \bar{\pi} \rangle}_{J^{(\pi)}}$ and (5.51) gives $\bar{\Delta}^{\langle \bar{\pi} \rangle}_{J^{(\pi)}} = \bar{\Delta}^{\langle \bar{\pi}+1 \rangle}_{J^{(\pi)}}$. The subspace selection condition (5.29) asserts $\Delta^{\langle \underline{\pi} \rangle}_{J^{(\pi)}} \geq \tau_1 \Delta^{\langle \underline{\pi} \rangle}$ and stopping condition (X1) implies $\Delta^{\langle \bar{\pi}+1 \rangle}_{J^{(\pi)}} \overset{(5.51)}{=} \Delta^{(\pi)}(\bar{w}^{\langle \bar{\pi}+1 \rangle}_{R_{J^{(\pi)}}}, \hat{y}^{\langle \bar{\pi}+1 \rangle}_{J^{(\pi)}}) < \tau_2 \Delta^{\langle \underline{\pi} \rangle}_{J^{(\pi)}}$.

This yields

$$\begin{aligned}\Delta^{\langle\bar\pi\rangle} - \Delta^{\langle\bar\pi+1\rangle} &= (\Delta^{\langle\bar\pi\rangle}_{J(\pi)} - \Delta^{\langle\bar\pi+1\rangle}_{J(\pi)}) + (\bar\Delta^{\langle\bar\pi\rangle}_{J(\pi)} - \bar\Delta^{\langle\bar\pi+1\rangle}_{J(\pi)}) + (\delta^{\langle\bar\pi\rangle}_{\bar J(\pi)} - \delta^{\langle\bar\pi+1\rangle}_{\bar J(\pi)}) \\ &= (\Delta^{\langle\underline\pi\rangle}_{J(\pi)} - \Delta^{\langle\bar\pi+1\rangle}_{J(\pi)}) + (\delta^{\langle\bar\pi\rangle}_{\bar J(\pi)} - \delta^{\langle\bar\pi+1\rangle}_{\bar J(\pi)}) \\ &\geq (1-\tau_2-\tau_3)\Delta^{\langle\underline\pi\rangle}_{J(\pi)} \\ &\geq \underbrace{\tau_1(1-\tau_2-\tau_3)}_{=:\tau\in(0,1)}\Delta^{\langle\underline\pi\rangle}.\end{aligned} \tag{5.52}$$

Note that this shows $\Delta^{\langle\bar\pi\rangle} - \Delta^{\langle\bar\pi+1\rangle} \geq 0$ for all $\bar\pi = \sigma \geq \underline\sigma$. Together with $\Delta^{\langle\underline\pi\rangle} = \Delta^{\langle\underline\pi+1\rangle}$ (see above) we get therefore that the sequence $(\Delta^{\langle\sigma\rangle})_{\sigma\geq\underline\sigma}$ is non-increasing.

Now assume $\bar\pi = \sigma > \bar\sigma$, then we have $\underline\pi \geq \underline\sigma$ and thus $\Delta^{\langle\underline\pi\rangle} \geq \Delta^{\langle\bar\pi\rangle}$. From (5.52) we obtain

$$\Delta^{\langle\bar\pi\rangle} - \Delta^{\langle\bar\pi+1\rangle} \geq \tau\Delta^{\langle\underline\pi\rangle} \geq \tau\Delta^{\langle\bar\pi\rangle}$$

and so

$$\Delta^{\langle\bar\pi+1\rangle} \leq (1-\tau)\Delta^{\langle\bar\pi\rangle}.$$

Together with the case $\sigma = \underline\pi$ above we get $\lim_{\sigma\in\mathbb{N}_0}\Delta^{\langle\sigma\rangle} = 0$, which completes the proof. □

Corollary 5.27 *If f is bounded from below and $\varepsilon = 0$, the predicted decrease $\Delta^{\langle\sigma\rangle} = f(\hat y^{\langle\sigma\rangle}) - \hat f_{\bar w^{\langle\sigma\rangle}}(\hat y^{\langle\sigma\rangle}) + \frac{1}{u}\|g(\bar x^{\langle\sigma\rangle})\|^2$ goes to zero for an appropriate subsequence $\Sigma^* \subseteq \Sigma$. In particular, $f(\hat y^{\langle\sigma\rangle}) - \hat f_{\bar w^{\langle\sigma\rangle}}(\hat y^{\langle\sigma\rangle})$ and $\|g(\bar x^{\langle\sigma\rangle})\|$ go to zero, too, for the subsequence Σ^*.*

Proof. Depending on whether an infinite number of descent steps occurs or not the claim follows either from Lemma 5.25 or Lemma 5.26. The last statement follows from the fact $f(\hat y^{\langle\sigma\rangle}) - \hat f_{\bar w^{\langle\sigma\rangle}}(\hat y^{\langle\sigma\rangle}) \geq 0$ and $\|g(\bar x^{\langle\sigma\rangle})\| \geq 0$. □

Theorem 5.28 *Suppose $\emptyset \neq \operatorname{Argmin} f$ is bounded. Then for an appropriate subsequence $\Sigma^* \subseteq \Sigma$ the sequences $(\hat y^{\langle\sigma\rangle})_{\sigma\in\Sigma^*}$ and $(\bar x^{\langle\sigma\rangle})_{\sigma\in\Sigma^*}$ that are generated by the parallel bundle algorithm have the following properties.*

(i) each accumulation point of $(\hat y^{\langle\sigma\rangle})_{\sigma\in\Sigma^*}$ is an optimal solution of (LD),

*(ii) each accumulation point of $(\bar x^{\langle\sigma\rangle})_{\sigma\in\Sigma^*}$ is an optimal solution of (*conv *P).*

Proof. Let $f^* := \min\{f(y)\colon y \in \mathbb{R}^M\}$. The boundedness of the level set $\{y\colon f(y) \le f^*\}$ implies the boundedness of all level sets, particularly of the set $\mathcal{S} := \{y\colon f(y) \le f(\hat{y}^{\langle 0\rangle})\}$. Because $(f(\hat{y}^{\langle\sigma\rangle}))_\sigma$ is non-increasing, see Lemma 5.24, we have $\hat{y}^{\langle\sigma\rangle} \in \mathcal{S}$ for all $\sigma \in \Sigma$ and therefore the sequence $(\hat{y}^{\langle\sigma\rangle})_\sigma$ is bounded. Likewise, $(\bar{l}^{\langle\sigma\rangle}, \bar{x}^{\langle\sigma\rangle})_\sigma$ lies in the compact set $\operatorname{conv} W$ by Lemma 5.22.

Let $\Sigma' \subseteq \Sigma$ be a subsequence that drives $\Delta^{\langle\sigma\rangle}$ to zero, according to Corollary 5.27. Let $(l^*, x^*), y^*$ be accumulation points of $(\bar{l}^{\langle\sigma\rangle}, \bar{x}^{\langle\sigma\rangle})_{\sigma\in\Sigma^*}$, $(\hat{y}^{\langle\sigma\rangle})_{\sigma\in\Sigma^*}$ for an appropriate subsequence $\Sigma^* \subseteq \Sigma$, then Corollary 5.27 asserts $g(x^*) = 0$ and therefore x^* is a feasible solution of (conv P). By Definition 3.1 (in Section 3.3) of $(\operatorname{conc} h_r)$ and W_r we have $\bar{l}_r \le (\operatorname{conc} h_r)(\bar{x}_r)$ for all $(\bar{l}_r, \bar{x}_r) \in \operatorname{conv} W_r$ $(r \in R)$, thus $f(\hat{y}^{\langle\sigma\rangle}) - \hat{f}_{(\bar{l}^{\langle\sigma\rangle}, \bar{x}^{\langle\sigma\rangle})}(\hat{y}^{\langle\sigma\rangle}) \to 0$ implies

$$
\begin{aligned}
(\operatorname{conc} h)(x^*) \quad &\overset{\sigma\in\Sigma^*}{\longleftarrow} \quad \sum_{r\in R} \Big[(\operatorname{conc} h_r)(\bar{x}_r^{\langle\sigma\rangle}) + \underbrace{(\hat{y}^{\langle\sigma\rangle})^T}_{\text{bounded}} \underbrace{g(\bar{x}_r^{\langle\sigma\rangle})}_{\to 0}\Big] \\
&\ge \sum_{r\in R} \Big[\bar{l}_r^{\langle\sigma\rangle} + (\hat{y}^{\langle\sigma\rangle})^T g(\bar{x}_r^{\langle\sigma\rangle})\Big] = \hat{f}_{(\bar{l}^{\langle\sigma\rangle}, \bar{x}^{\langle\sigma\rangle})}(\hat{y}^{\langle\sigma\rangle}) \quad \overset{\sigma\in\Sigma^*}{\longrightarrow} \quad f(y^*).
\end{aligned}
$$

Thus x^* is an optimal solution of (conv P) and y^* is an optimal solution of (LD). □

5.4 Extension to Stronger Coupling

In many applications the assumption that R_j is small for most $j \in M$ is actually too strong. Consider, *e. g.*, a constraint for a common resource ensuring that only a limited number of all objects may make use of this resource at a specific point in time. Even though such a constraint $j \in M$ couples many subproblems, it typically influences but a few of them, because most objects need this resource at some other time. A typical example for this situation are the station capacity constraints of the train timetabling problem (see (2.6) in Chapter 2).

This section covers an extension of the basic parallel bundle method presented in Section 5.3 to stronger coupled problems. In this approach the structure described above is exploited by keeping track of those constraints j and subproblems $r \in R$, that have proved to interact for at least one feasible solution $(\bar{x}^{\langle\sigma\rangle}, \hat{y}^{\langle\sigma\rangle})$ up to the current marker σ, and the optimisation process is restricted to these.

The structure of this section will follow the same outline as Section 5.3. In order to clearly discern the new objects of this extended version from the objects of the last section, we will use superscript $\{\sigma\}$ for index markers and $[\pi]$ for objects belonging to a process π. The algorithm will have the same structure as the standard version described in Section 5.3.3. In particular, we will use the same notation for processes as before: each process π is associated with two index markers $\underline{\pi}$ and $\bar{\pi}$ referring to the global state when π starts resp. finishes its computation. For each $\sigma \in \mathbb{N}_0 \cup \{\infty\}$ we arrange the processes

in the following groups

$$\underline{\Pi}^{\{\sigma\}} := \{\pi = (\underline{\pi}, \bar{\pi})\colon \underline{\pi} < \bar{\pi} \text{ and } \underline{\pi} < \sigma\}, \qquad \underline{\Pi}^{\{\sigma\}}$$

$$\bar{\Pi}^{\{\sigma\}} := \{\pi = (\underline{\pi}, \bar{\pi})\colon \underline{\pi} < \bar{\pi} < \sigma\}, \qquad \bar{\Pi}^{\{\sigma\}}$$

$$\Pi^{\{\sigma\}} := \{\pi = (\underline{\pi}, \bar{\pi})\colon \underline{\pi} < \sigma \leq \bar{\pi}\} = \underline{\Pi}^{\{\sigma\}} \setminus \bar{\Pi}^{\{\sigma\}}. \qquad \Pi^{\{\sigma\}}$$

The groups are completely analogue to the standard case. Furthermore Observation 5.10 holds for these sets, too. The set of all index markers $\sigma \in \mathbb{N}_0$ visited by the algorithm is

$$\Sigma := \{0\} \cup \{\sigma \in \mathbb{N}\colon \Pi^{\langle\sigma\rangle} \neq \Pi^{\langle\sigma-1\rangle}\}.$$

5.4.1 Introduction

In order to prevent the structure of the constraints from implying large subspaces, the algorithm maintains sets $J_r^{\{\sigma\}} \subset J_r$ for $r \in R$. These sets collect the indexes of all those constraints acting on subproblem r, whose Lagrange multipliers presumably influence the optimal solution of $f_r(y)$ or its value. In the corresponding restricted subproblems the other constraints $J_r \setminus J_r^{\{\sigma\}}$ will be ignored in the following sense.

When a process π selects its subspace $J^{[\pi]} \subseteq M$ and corresponding subproblems $R^{[\pi]} \subseteq R$ at $\sigma = \underline{\pi}$, it assumes that only the multipliers belonging to $M^{[\pi]} := \bigcup_{r \in R^{[\pi]}} J_r^{\{\underline{\pi}\}}$ have an influence on the solution of these subproblems. In other words, the subprocess treats the problem as if all entries $C_{J_r \setminus J_r^{\{\underline{\pi}\}}}$ equal zero. See Figure 5.2 for an illustration of the constraint matrix w. r. t. *activity sets* $J_r^{\{\underline{\pi}\}}$ when the subspace of π is selected at $\underline{\pi}$. The difficulty arises if this assumption is no longer true when the process finishes its work at $\sigma = \bar{\pi}$. At this point the process will not only need to include newly discovered influences due to new optimal solutions $x_r \in \mathfrak{X}_r$ for some $r \in R^{[\pi]}$ in updated sets $J_r^{\{\bar{\pi}+1\}}$, but possibly also encounter modified sets $J_r^{\{\bar{\pi}\}} \neq J_r^{\{\underline{\pi}\}}$ for some $r \in R^{[\pi]}$ due to changes of the Lagrange multipliers in $M \setminus M^{[\pi]}$ by other processes. These changes might or might not invalidate the results of π. In the former case these results have to be discarded and the dependency information between subproblems and constraints, *i. e.* the sets $J_r^{\{\sigma\}}$, $r \in R^{[\pi]}$, has to be updated. Thus an important aspect of the algorithm are conditions that ensure that the results computed by a process π remain valid, which we will study next.

5.4.2 Dependencies between Constraints and Subproblems

As stated in the previous section, the dependencies between constraints and subspaces are represented by sets $J_r^{\{\sigma\}} \subseteq J_r$, $r \in R$. Most parts of the algorithm work as if only the dependencies $J_r^{\{\sigma\}}$ existed. During the run of the algorithm these sets may be enlarged if further dependencies are detected. This means that the set of all constraints $J_r^{\{\sigma\}}$ may *change* during the algorithm (in contrast to the static subspace J_r). Because of this, we will often call the subspace $J_r^{\{\sigma\}}$ the *activity set* of $r \in R$ at index $\sigma \in \mathbb{N}_0$. *activity set*

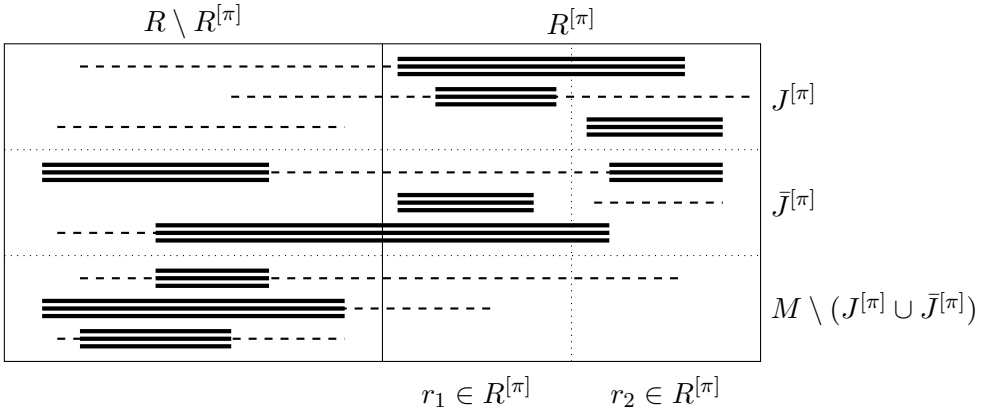


Figure 5.2: The thick lines indicate coefficients of the constraint matrix that are in the current activity sets $J_r^{\{\sigma\}}$, the dashed lines are coefficients that are currently not in these sets. A subspace $J \subseteq M$ divides the subproblems R into two parts: the subproblems R_J, which *actually* interact with J, and $R \setminus R_J$, which do not *actually* interact with J. The constraint $\bar{J}$ are those that *actually* interact with both, R_J and $R \setminus R_J$. A subspace problem may detect new interactions, *i. e.*, some parts of the dashed lines may turn thick because the activity sets grow. The algorithm must handle all these cases, that may influence the partition induces by J.

Having changing activity sets also means that we have to consider slightly changed objects and relations compared with the standard algorithm. When a process π starts, it selects its subspace at $\underline{\pi}$ and sets up its subproblem w. r. t. the current global dependency information (both, the constraint dependencies described by the dependency graph, and the subproblem-constraint dependencies described by the sets $J_r^{\{\sigma\}}$, $r \in R$). In particular, we define the following sets w. r. t. a process π resp. its start index $\underline{\pi}$:

$$R_j^{[\pi]} := \{r \in R \colon j \in J_r^{\{\underline{\pi}\}}\}, \qquad \text{the subproblems interacting at } \underline{\pi} \text{ with } j, \tag{5.53}$$

$$R_J^{[\pi]} := \textstyle\bigcup_{j \in J} R_j^{[\pi]}, \qquad \text{the subproblems interacting at } \underline{\pi} \text{ with } J, \tag{5.54}$$

and specific for the process π

$$R^{[\pi]} := R_{J^{[\pi]}}^{[\pi]}, \qquad \text{subproblems associated with process } \pi \tag{5.55}$$

$$J_r^{[\pi]} := J_r^{\{\underline{\pi}\}} \cap J^{[\pi]}, \qquad \text{dual variables in } J^{[\pi]} \text{ interacting with } r \text{ at } \underline{\pi}, \tag{5.56}$$

$$\bar{J}_r^{[\pi]} := J_r^{\{\underline{\pi}\}} \setminus J^{[\pi]}, \qquad \text{dual variables not in } J^{[\pi]} \text{ interacting with } r \text{ at } \underline{\pi}, \tag{5.57}$$

$$\bar{J}^{[\pi]} := \textstyle\bigcup_{r \in R^{[\pi]}} \bar{J}_r^{[\pi]}, \qquad \text{dual variables not in } J^{[\pi]} \text{ interacting with } R^{[\pi]} \text{ at } \underline{\pi}, \tag{5.58}$$

$$\mathcal{X}^{[\pi]} := \textstyle\bigtimes_{r \in R^{[\pi]}} \mathcal{X}_r, \qquad \text{primal ground set associated with } R^{[\pi]}.$$

These definitions imply the following relations.

Observation 5.29 *It holds*

$$J_r^{\{\underline{\pi}\}} = J_r^{[\pi]} \cup \bar{J}_r^{[\pi]} \subseteq J^{[\pi]} \cup \bar{J}^{[\pi]}, \qquad \text{for all } r \in R^{[\pi]}, \tag{5.59}$$

$$J_r^{\{\underline{\pi}\}} \cap J^{[\pi]} = \emptyset, \qquad \text{for all } r \in R \setminus R^{[\pi]}, \tag{5.60}$$

$$J_r^{\{\sigma\}} \subseteq J_r, \qquad \text{for all } r \in R, \sigma \in \mathbb{N}_0. \tag{5.61}$$

Proof. Directly from the definitions. □

Note that if $J_r^{\{\underline{\pi}\}} = J_r$ for all $r \in R$, then these objects correspond exactly to the objects used in the standard parallel bundle algorithm.

Observation 5.30 *Assume $J_r^{\{\underline{\pi}\}} = J_r$ for all $r \in R$, then the following relations hold.*

$$R_j^{[\pi]} = R_j, \qquad R^{[\pi]} = R_{J^{[\pi]}},$$
$$J_r^{[\pi]} = J_r \cap J^{[\pi]},$$
$$\bar{J}_r^{[\pi]} = J_r \setminus J^{[\pi]},$$
$$\mathfrak{X}^{[\pi]} = \mathfrak{X}_{R_{J^{[\pi]}}}.$$

Proof. Directly from the definitions. □

Next we look at the global expected progress $\Delta(\bar{w}, \hat{y})$ for $\bar{w} = (\bar{l}, \bar{x}) \in \operatorname{conv} W$, $\hat{y} \in \mathbb{R}^M$. Analogous to Observation 5.8 we split this expected progress into three parts. But this time this splitting does not only depend on the selected subspace J but also on the coupling between the subproblems and constraints at some index $\sigma \in \mathbb{N}_0$. Note that we split w. r. t. the coupling at $\bar{\pi} + 1$ (the sets $J_r^{\{\bar{\pi}+1\}}$), *i. e.* the global data after π has just updated the global data with its results. The reason why we choose this information and not the coupling at $\underline{\pi}$ will become apparent later.

$$\hat{J}^{[\pi]} := \textstyle\bigcup_{r \in R^{[\pi]}} J_r^{\{\bar{\pi}+1\}} \setminus J^{[\pi]}, \tag{5.62}$$

$$\Delta_{\pi,J}(\bar{w}, \hat{y}) := \sum_{r \in R_J^{[\pi]}} \left[f_r(\hat{y}) - \hat{f}_{\bar{w}_r, r}(\hat{y}) \right] + \frac{1}{u} \|g(\bar{x})_J\|^2, \tag{5.63}$$

$$\Delta_\pi(\bar{w}, \hat{y}) := \Delta_{\pi, J^{[\pi]}}(\bar{w}, \hat{y}), \tag{5.64}$$

$$\bar{\Delta}_\pi(\bar{w}, \hat{y}) := \sum_{r \in R \setminus R^{[\pi]}} \left[f_r(\hat{y}) - \hat{f}_{\bar{w}_r, r}(\hat{y}) \right] + \frac{1}{u} \|g(\bar{x})_{M \setminus (J^{[\pi]} \cup \hat{J}^{[\pi]})}\|^2, \tag{5.65}$$

$$\delta_\pi(\bar{w}) := \frac{1}{u} \|g(\bar{x}_r)_{\hat{J}^{[\pi]}}\|^2.$$

Remark 5.31 Note that only $\bar{\Delta}_\pi$ and δ_π depend on $\hat{J}^{[\pi]}$ (and thus on global data at $\bar{\pi}+1$). Δ_π does *not* depend on $\hat{J}^{[\pi]}$.

Observation 5.32 *(see Observation 5.8)*
For each process π, each $\bar{w} = (\bar{l}, \bar{x}) \in \operatorname{conv} W$ and $\hat{y} \in \mathbb{R}^M$ there holds

$$\Delta(\bar{w}, \hat{y}) = \Delta_\pi(\bar{w}, \hat{y}) + \bar{\Delta}_\pi(\bar{w}, \hat{y}) + \delta_\pi(\bar{x}).$$

Proof. Direct computation. □

5.4.3 Conditions of Global Validity for Restricted Subproblems

In this section we give and discuss some sufficient properties for the validity of the results computed by some process. More precisely, we state some conditions that guarantee that optimal solutions that have been computed under certain dependency assumptions, *i. e.* sets $J_r^{\{\sigma\}}$, $r \in R^{[\pi]}$, are indeed correct optimal solutions of the original problem.

consistent

Definition 5.33 Given $r \in R$, a multiplier vector $y \in \mathbb{R}^M$, a point $\hat{x}_r \in \mathcal{X}_r$, a point $\bar{x}_r \in \operatorname{conv} \mathcal{X}_r$ and a set $J'_r \subseteq J_r$. The 4-tuple $(y, \hat{x}_r, \bar{x}_r, J'_r)$ is called *consistent* for r if

$$y_j \cdot (C_{j,r} x_r) \geq 0, \qquad \text{for all } x_r \in \mathcal{X}_r, j \in J_r \setminus J'_r, \tag{C1}$$
$$C_{M \setminus J'_r, r} \hat{x}_r = 0, \tag{C2}$$
$$C_{M \setminus J'_r, r} \bar{x}_r = 0, \tag{C3}$$
$$\hat{x}_r \in \operatorname{Argmax}\left\{ h_r(x_r) - (y_{J'_r})^T C_{J'_r, r} x_r \colon x_r \in \mathcal{X}_r \right\}. \tag{C4}$$

We say the (global) data at σ is consistent if $(\hat{y}^{\{\sigma\}}, \hat{x}_r^{\{\sigma\}}, \bar{x}_r^{\{\sigma\}}, J_r^{\{\sigma\}})$ is consistent for each $r \in R$.

The intuition behind consistency is as follows. Given that the global data at σ is consistent, then (C1)–(C4) ensure that the points $\hat{x}_r^{\{\sigma\}}$, $r \in R$, are indeed optimal solutions when the remaining indexes $J_r \setminus J_r^{\{\sigma\}}$ of this $\hat{y}^{\{\sigma\}}$ are included as well, *i. e.*,

$$\hat{x}_r^{\{\sigma\}} \in \operatorname{Argmax}\left\{ h_r(x_r) - (\hat{y}^{\{\sigma\}})^T C_{\bullet, r} x_r \colon x_r \in \mathcal{X}_r \right\}.$$

Similarly, condition (C3) guarantees that the global affine minorant associated with $\bar{w}^{\{\sigma\}} = (\bar{l}^{\{\sigma\}}, \bar{x}^{\{\sigma\}})$ only depends on the restricted subspace $J_r^{\{\sigma\}}$. This will be made rigorous below.

Remark 5.34 Condition (C1) is usually not checked explicitly but guaranteed by exploiting the problem structure. For example, if $C_{j,r} \geq 0$ and $x_r \geq 0$ for all $x_r \in \mathcal{X}_r$ (which is often the case for combinatorial problems) (C1) holds whenever $\hat{y}_j^{\{\sigma\}} \geq 0$.

Remark 5.35 Note that $(\hat{y}, \hat{x}_r, \bar{x}_r, J_r)$, $r \in R$, is consistent independent of the choice of $\hat{y}$ and $\bar{x}_r$ if $\hat{x}_r$ fulfils (C4), *i. e.*, choosing maximal activity sets $J_r^{\{\sigma\}} = J_r$ would be sufficient to get consistency.

In the algorithm we will make use of the following weaker condition.

Definition 5.36 Given $r \in R$, a multiplier vector $y \in \mathbb{R}^M$, a point $\hat{x}_r \in \mathcal{X}_r$, a point $\bar{x}_r \in \operatorname{conv} \mathcal{X}_r$ and a set $J_r' \subseteq J_r$. The 4-tuple $(y, \hat{x}_r, \bar{x}_r, J_r')$ is called *weakly consistent* if it satisfies (C1), (C4) and *weakly consistent*

$$y_j \cdot (C_{j,r}\hat{x}_r) = 0, \qquad \text{for all } j \in M \setminus J_r', \tag{C2'}$$
$$y_j \cdot (C_{j,r}\bar{x}_r) = 0, \qquad \text{for all } j \in M \setminus J_r'. \tag{C3'}$$

Obviously, consistency implies weak consistency, but the converse will not be true in general.

Observation 5.37 *Let $r \in R$ and $\hat{y} \in \mathbb{R}^M$, $\hat{x}_r \in \mathcal{X}_r$, $\bar{x}_r \in \operatorname{conv} \mathcal{X}_r$ and $J_r' \subseteq J_r$, so that $(\hat{y}, \hat{x}_r, \bar{x}_r, J_r')$ is consistent, then it is also weakly consistent.*

Proof. (Ci) implies (Ci') for $i \in \{2, 3\}$. □

An important observation is that (weak) consistency remains true if only the subspaces J_r' are enlarged.

Observation 5.38 *Given $r \in R$, $y \in \mathbb{R}^M$, $\hat{x}_r \in \mathcal{X}_r$, $\bar{x}_r \in \operatorname{conv} \mathcal{X}_r$ and sets $J_r' \subseteq J_r'' \subseteq J_r$, suppose $(y, \hat{x}_r, \bar{x}_r, J_r')$ is (weakly) consistent. Then $(y, \hat{x}_r, \bar{x}_r, J_r'')$ is (weakly) consistent, too. In particular, $\hat{x}_r$ is an optimal solution for $\max L_r(\cdot, y)$ with*

$$f_r(y) = L_r(\hat{x}_r, y) = h_r(\hat{x}_r) - (y_{J_r'})^T C_{J_r',r}\hat{x}_r. \tag{5.66}$$

Proof. Conditions (C1)–(C3), (C2') and (C3') are obvious, it remains to show (C4). Given $r \in R$ and let $x_r \in \mathcal{X}_r$, be an arbitrary primal point. Then

$$\begin{aligned} h_r(x_r) - (y_{J_r''})^T C_{J_r'',r} x_r &= h_r(x_r) - (y_{J_r'})^T C_{J_r',r} x_r - \underbrace{(y_{J_r''\setminus J_r'})^T C_{J_r''\setminus J_r',r} x_r}_{\geq 0 \text{ by (C1)}} \\ &\leq h_r(x_r) - (y_{J_r'})^T C_{J_r',r} x_r \\ &\overset{(C4)}{\leq} h_r(\hat{x}_r) - (y_{J_r'})^T C_{J_r',r} \hat{x}_r \\ &= h_r(\hat{x}_r) - (y_{J_r'})^T C_{J_r',r} \hat{x}_r - \underbrace{(y_{J_r''\setminus J'})^T C_{J_r''\setminus J',r} \hat{x}_r}_{=0 \text{ as (C2') holds for } \hat{x}_r}. \end{aligned}$$

Therefore $\hat{x}_r$ is an optimal solution for the subproblem induced by J_r'' with the same objective value. Putting $J_r'' = J_r$, (5.66) follows from the definitions (5.10), (5.14) and (5.15). □

5.4.4 Subspace Selection

Analogous to the standard parallel bundle method Section 5.3.5 the first step is the selection of an appropriate subspace. One important difference is that we do not have a result analogous to Observation 5.11. This means that it is *not* sufficient to enforce disjoint sets of subproblems for parallel processes in order to get disjoint subspaces as well. Instead we have to *enforce* disjoint subspaces by maintaining a set of blocked constraints $B_M^{\{\sigma\}} \subseteq M$ (analogously to the set of blocked subproblems $B^{\{\sigma\}}$).

blocked constraints $B_M^{\{\sigma\}}$

The aim of the subspace selection is the same as for the standard case. We have to find an unblocked subspace that provides sufficient decrease compared with the global expected progress, *i. e.* for some process π we must guarantee

$$\Delta_\pi(\bar{w}, \hat{y}) \geq \tau_1 \Delta(\bar{w}, \hat{y}). \tag{5.67}$$

Analogously to Section 5.3.4 we can now proof that such a subspace can always be found.

Observation 5.39 *(see Observation 5.12)*
For subspaces $J \subseteq J' \subseteq M$ and $\bar{w} = (\bar{l}, \bar{x}) \in \operatorname{conv} W$, $\hat{y} \in \mathbb{R}^M$ there holds

$$\Delta_{\pi,J}(\bar{w}, \hat{y}) \leq \Delta_{\pi,J'}(\bar{w}, \hat{y}). \tag{5.68}$$

Furthermore, condition (5.67) holds for $J = J^{[\pi]} = M$.

Proof. Analogous to the proof of (5.30) and using $\Delta_{\pi,J^{[\pi]}}(\bar{w}, \hat{y}) = \Delta_\pi(\bar{w}, \hat{y})$ by definition (5.64). □

Because the definition of Δ_π depends on the activity sets $J_r^{\{\pi\}}$ associated with process π when it starts, we are not able to prove a result analogous to Lemma 5.13, yet. Instead we defer this result to Section 5.4.8 (Lemma 5.53).

Now we can state the subspace selection procedure for the extended version. This procedure differs from the standard version (Algorithm 5.2) in the following two aspects:

1. It uses the active sets $J_r^{\{\sigma\}}$ instead of J_r for all $r \in R$ to define the subproblem,
2. it has to observe the set of blocked constraints B_M.

Algorithm 5.6: EXT-SELECTSUBSPACE

Input : global data at $\underline{\pi}$
Output : **TRUE** if a subspace can be selected
Changes: sets $J^{[\pi]}$
$X \leftarrow M \setminus B_M^{\{\underline{\pi}\}}$, $J^{[\pi]} \leftarrow \emptyset$
while $J^{[\pi]}$ *does not satisfy (5.67) and* $X \neq \emptyset$ **do**
  select $j \in \operatorname{Argmax}\{\Delta_{\pi,\{j\}}(\bar{w}^{\{\underline{\pi}\}}, \hat{y}^{\{\underline{\pi}\}}) \colon j \in X\}$
  $Y \leftarrow \{j\} \cup \{j' \in M \colon (j,j') \in E^{\{\underline{\pi}\}}\}$
  $\bar{Y} \leftarrow \bigcup_{r \in R_Y^{[\pi]}} \bar{J}_r^{[\pi]}$
  if $(Y \cup \bar{Y}) \cap B_M^{\{\underline{\pi}\}} = \emptyset$ *and* $R_Y^{[\pi]} \cap B^{\{\underline{\pi}\}} = \emptyset$ **then**
1   $J^{[\pi]} \leftarrow J^{[\pi]} \cup Y$, $X \leftarrow X \setminus Y$
  else
    $X \leftarrow X \setminus \{j\}$
if $J^{[\pi]}$ *satisfies (5.67)* **then**
  return *TRUE*
else
  $J^{[\pi]} \leftarrow \emptyset$
  return *FALSE*

Observation 5.40 *(see Observation 5.14)*
If Algorithm 5.6 returns a subspace $\emptyset \neq J^{[\pi]} \subseteq M$*, then* $J^{[\pi]}$ *fulfils (5.67) and*

$$(J^{[\pi]} \cup \bar{J}^{[\pi]}) \cap B_M^{\{\underline{\pi}\}} = \emptyset = R^{[\pi]} \cap B^{\{\underline{\pi}\}}. \tag{5.69}$$

If the algorithm is called without blocked subproblems or constraints, i. e. $B^{\{\underline{\pi}\}} = \emptyset = B_M^{\{\underline{\pi}\}}$*, then the returned subspace is not empty, i. e.* $J^{[\pi]} \neq \emptyset$*.*

Proof. The algorithm is finite because at least one element is removed from X in each iteration. The validity of (5.67) follows from the final test. The second statement follows inductively: $J^{[\pi]} = \emptyset$ fulfils (5.69) trivially. Define the notation

$$\mathcal{F}(J) := \bigcup_{r \in R_J^{[\pi]}} J_r^{\{\pi\}} \setminus J$$

for $J \subseteq M$ and observe that $\mathcal{F}(J \cup J') \subseteq \mathcal{F}(J) \cup \mathcal{F}(J')$ because $R_{J \cup J'}^{[\pi]} = R_J^{[\pi]} \cup R_{J'}^{[\pi]}$. Note that $\bar{J}^{[\pi]} = \mathcal{F}(J^{[\pi]})$ by (5.58) and in each iteration $\bar{Y} = \mathcal{F}(Y)$. Before $J^{[\pi]}$ is enlarged in A5.6.1 there holds

$$\begin{aligned} &((J^{[\pi]} \cup Y) \cup \mathcal{F}(J^{[\pi]} \cup Y)) \cap B_M^{\{\pi\}} \\ &\subseteq ((J^{[\pi]} \cup \mathcal{F}(J^{[\pi]})) \cap B_M^{\{\pi\}}) \cup ((Y \cup \mathcal{F}(Y)) \cap B_M^{\{\pi\}}) = \emptyset, \end{aligned}$$

and

$$R_{J^{[\pi]} \cup Y}^{[\pi]} \cap B^{\{\pi\}} = (R_{J^{[\pi]}}^{[\pi]} \cap B^{\{\pi\}}) \cup (R_Y^{[\pi]} \cap B^{\{\pi\}}) = \emptyset.$$

The respective first terms are empty by induction hypothesis, the latter terms because of the previous test.

The last statement follows because if $B^{\{\pi\}} = \emptyset = B_M^{\{\pi\}}$ then $J^{[\pi]}$ will eventually become M, which fulfils (5.67) by Observation 5.39. □

5.4.5 The Subspace Problem

The subspace problem is very similar to the standard case. It differs only in the fact that for each $r \in R$ only the coefficients of the constraint Matrix $C_{J_r^{\{\pi\}},r}$ are considered instead of $C_{J_r,r}$. In other words, the subproblem is built as if $C_{J \setminus J_r^{\{\pi\}},r}$ were equal to zero. Therefore, the subspace problem itself differs only slightly from the standard case. But the connection between the subspace problem and the global problem gets more complicated, because the global problem (*i. e.* the original optimisation problem that we want to solve) always uses the full constraint matrix.

The data that defines the subspace problem is

(i) the associated subspace, $J^{[\pi]} \subseteq M$,

(ii) primal aggregate minorant when the process starts, $\bar{w}^{\{\pi\}} \in \operatorname{conv} W$,

(iii) initial centre of stability when the process starts, $\hat{y}^{\{\pi\}} \in \mathbb{R}^M$,

(iv) activity sets when the process starts, $J_r^{\{\pi\}}$, $r \in R$.

We start again with the description of the subspace problem by incorporating the influence of the fixed part of y into the objective function while ignoring everything contained in sets $J_r \setminus J_r^{\{\pi\}}$

$$c_r^{[\pi]} := -(C_{\bar{J}_r^{[\pi]},r})^T \hat{y}_{\bar{J}_r^{[\pi]}}^{\{\pi\}}, \qquad \text{for all } r \in R^{[\pi]}, \tag{5.70}$$

$$h_r^{[\pi]}(x_r) := h_r(x_r) + (c_r^{[\pi]})^T x_r, \qquad \text{for all } x_r \in \mathcal{X}_r, r \in R^{[\pi]}. \tag{5.71}$$

The Lagrangian and the dual functions are

$$L_r^{[\pi]}(x_r, y^{[\pi]}) := h_r^{[\pi]}(x_r) - (y^{[\pi]}_{J_r^{[\pi]}})^T C_{J_r^{[\pi]},r} x_r, \quad x_r \in \mathcal{X}_r, r \in R^{[\pi]}, y^{[\pi]} \in \mathbb{R}^{J^{[\pi]}}, \tag{5.72}$$

$$f_r^{[\pi]}(y^{[\pi]}) := \max_{x_r \in \mathcal{X}_r} L_r^{[\pi]}(x_r, y^{[\pi]}), \quad r \in R^{[\pi]}, y^{[\pi]} \in \mathbb{R}^{J^{[\pi]}}, \tag{5.73}$$

$$f^{[\pi]}(y^{[\pi]}) := b^T_{J^{[\pi]}} y^{[\pi]} + \sum_{r \in R^{[\pi]}} f_r^{[\pi]}(y^{[\pi]}). \tag{5.74}$$

The subspace problem then reads

$$\text{(ExtSub}(\pi)\text{)} \qquad \begin{aligned} \text{minimise} \quad & f^{[\pi]}(y^{[\pi]}) \\ \text{subject to} \quad & y^{[\pi]} \in \mathbb{R}^{J^{[\pi]}}. \end{aligned}$$

As for the simple parallel bundle algorithm, the subspace problem will be solved by a (classical) bundle method, and we introduce linear minorants and cutting plane models for the objective functions $f_r^{[\pi]}(y^{[\pi]})$, $r \in R^{[\pi]}$, and $f^{[\pi]}(y^{[\pi]})$. We collect all feasible primal solutions in

$$W^{[\pi]} := \bigtimes_{r \in R^{[\pi]}} W_r. \tag{$W^{[\pi]}$}$$

Each point $w^{[\pi]} = (l^{[\pi]}, x^{[\pi]}) \in \operatorname{conv} W^{[\pi]}$ defines a minorant of $f^{[\pi]}(y^{[\pi]})$,

$$\hat{f}^{[\pi]}_{w_r^{[\pi]},r}(y^{[\pi]}) := l_r^{[\pi]} + (c_r^{[\pi]})^T x_r^{[\pi]} - (y^{[\pi]}_{J_r^{[\pi]}})^T C_{J_r^{[\pi]},r} x_r^{[\pi]} \le f_r^{[\pi]}(y^{[\pi]}), \tag{5.75}$$

$$\hat{f}^{[\pi]}_{w^{[\pi]}}(y^{[\pi]}) := b^T_{J^{[\pi]}} y^{[\pi]} + \sum_{r \in R^{[\pi]}} \hat{f}^{[\pi]}_{w_r^{[\pi]},r}(y^{[\pi]}) \le f^{[\pi]}(y^{[\pi]}).$$

Due to the selective use of rows of C in (5.72), the gradient of the affine function $\hat{f}^{[\pi]}_{w^{[\pi]}}$ is somewhat clumsy to state,

$$g^{[\pi]}(x^{[\pi]}) := b_{J^{[\pi]}} - \Bigg[\sum_{r \in R_j^{[\pi]}} C_{j,r} x_r^{[\pi]} \Bigg]_{j \in J^{[\pi]}}. \tag{5.76}$$

In contrast to the standard setting, the difference between the subspace objects $L_r^{[\pi]}(x_r, y^{[\pi]})$, $\hat{f}^{[\pi]}(y^{[\pi]})$, $f^{[\pi]}(y^{[\pi]})$ is not only the additional term $(c_r^{[\pi]})^T x_r$ in the objective functions $h^{[\pi]}(x_r)$. In addition, the Lagrangian function is only defined in terms of the constraint matrix restricted to the rows $J_r^{[\pi]} = J_r^{\{\pi\}} \cap J^{[\pi]}$ instead of the full space $J^{[\pi]}$. Note that in the standard setting using the rows $J^{(\pi)}$ is the same as using $J_r \cap J$ because $C_{J \setminus J_r, r} = 0$ by the definition of J_r. Nevertheless we get some direct relations between the subspace and the global variants, but this time they involve the activity sets $J_r^{\{\pi\}}$.

Observation 5.41 *Given a process π, $r \in R^{[\pi]}$ and $y \in \mathbb{R}^M$ with $y_{\bar{J}_r^{[\pi]}} = \hat{y}^{\{\pi\}}_{\bar{J}_r^{[\pi]}}$, there hold*

$$L_r^{[\pi]}(x_r, y_{J^{[\pi]}}) = h_r(x_r) - (y_{J_r^{\{\pi\}}})^T C_{J_r^{\{\pi\}},r} x_r, \qquad \text{for } x_r \in \mathcal{X}_r, \tag{5.77}$$

$$\hat{f}^{[\pi]}_{w_r,r}(y_{J^{[\pi]}}) = l_r - (y_{J_r^{\{\pi\}}})^T C_{J_r^{\{\pi\}},r} x_r, \qquad \text{for } w_r \in \operatorname{conv} W_r. \tag{5.78}$$

Proof. Due to $y_{\bar{J}_r^{[\pi]}} = \hat{y}^{\{\pi\}}_{\bar{J}_r^{[\pi]}}$ we have $c_r^{[\pi]} = -C^T_{\bar{J}_r^{[\pi]},r} y_{\bar{J}_r^{[\pi]}}$. Furthermore (5.56) and (5.57) imply $J_r^{\{\pi\}} = J_r^{[\pi]} \dot\cup \bar{J}_r^{[\pi]}$. Hence, definitions (5.71) and (5.72) prove (5.77) and definition (5.75) shows (5.78). □

The following observation is central for the correctness of the algorithm. It states that in the case of weak consistency, the subspace problem objects correspond directly to the global objects.

Observation 5.42 *Given a process π, $r \in R^{[\pi]}$, $y \in \mathbb{R}^M$ with $y_{\bar{J}_r^{[\pi]}} = \hat{y}^{\{\pi\}}_{\bar{J}_r^{[\pi]}}$, $\hat{x}_r \in \mathcal{X}_r$ and $\bar{w}_r = (\bar{l}_r, \bar{x}_r) \in \operatorname{conv} W_r$, suppose $(y, \hat{x}_r, \bar{x}_r, J_r^{\{\pi\}})$ is weakly consistent. Then*

$$f_r^{[\pi]}(y_{J^{[\pi]}}) = L_r^{[\pi]}(\hat{x}_r, y_{J^{[\pi]}}) = L_r(\hat{x}_r, y) = f_r(y), \tag{5.79}$$

and

$$\hat{f}^{[\pi]}_{\bar{w}_r,r}(y_{J^{[\pi]}}) = \hat{f}_{\bar{w}_r,r}(y). \tag{5.80}$$

Proof. Together with

$$f_r^{[\pi]}(y_{J^{[\pi]}}) \overset{(5.73)}{=} \max_{x_r \in \mathcal{X}_r} L_r^{[\pi]}(x_r, y_{J^{[\pi]}}) \overset{(5.77),(C4)}{=} h_r(\hat{x}_r) - (y_{J_r^{\{\pi\}}})^T C_{J_r^{\{\pi\}},r} \hat{x}_r$$
$$\overset{(5.77)}{=} L_r^{[\pi]}(\hat{x}_r, y_{J^{[\pi]}})$$

(5.79) is a direct consequence of Observation 5.38 applied to the third expression. (5.80) follows by

$$\hat{f}^{[\pi]}_{\bar{w}_r,r}(y_{J^{[\pi]}}) \overset{(5.78)}{=} l_r - (y_{J_r^{\{\pi\}}})^T C_{J_r^{\{\pi\}},r} \bar{x}_r \overset{(C3')}{=} l_r - y^T C_{\bullet,r} \bar{x}_r \overset{(5.20)}{=} \hat{f}_{\bar{w}_r,r}(y).$$ □

Observation 5.43 *Given a process π and an $x \in \operatorname{conv} \mathcal{X}$ satisfying*

$$C_{j,r} x_r = 0 \quad \text{for } r \in R_j \setminus R_j^{[\pi]}, j \in J^{[\pi]}, \tag{5.81}$$

there holds

$$g^{[\pi]}(x_{R^{[\pi]}}) = g(x)_{J^{[\pi]}}. \tag{5.82}$$

Proof. We use $C_{j,R\setminus R_j} = 0$ for all $j \in M$ (see the definition (5.11) of R_j),

$$\begin{aligned} g^{[\pi]}(x_{R^{[\pi]}}) &\overset{(5.76)}{=} b_{J^{[\pi]}} - \left[\sum_{r \in R_j^{[\pi]}} C_{j,r} x_r + \sum_{r \in R \setminus R_j^{[\pi]}} \underbrace{C_{j,r} x_r}_{\overset{(5.81)}{=} 0} \right]_{j \in J^{[\pi]}} \\ &= b_{J^{[\pi]}} - C_{J^{[\pi]},\bullet} x \overset{(5.18)}{=} g(x)_{J^{[\pi]}} \end{aligned}$$ □

The bundle method solving the subspace problem is again a classical bundle method. Given a centre of stability $\hat{y}^{[\pi]} \in \mathbb{R}^{J^{[\pi]}}$ and an aggregate minorant $\bar{w}^{[\pi]} = (\bar{l}^{[\pi]}, \bar{x}^{[\pi]}) \in \operatorname{conv} W^{[\pi]}$, the next candidate point is

$$\bar{y}^{[\pi]} := \hat{y}^{[\pi]} - \frac{1}{u} g^{[\pi]}(\bar{x}^{[\pi]}).$$

The candidate is the minimiser of the augmented model

$$\bar{y}^{[\pi]} = \operatorname{argmin} \left\{ \hat{f}^{[\pi]}_{\bar{w}^{[\pi]}}(y^{[\pi]}) + \frac{u}{2} \|y^{[\pi]} - \hat{y}^{[\pi]}\|^2 \right\}.$$

The algorithm performs a descent step if the actual decrease $f^{[\pi]}(\hat{y}^{[\pi]}) - f^{[\pi]}(\bar{y}^{[\pi]})$ is sufficiently large in comparison to the predicted decrease

$$\begin{aligned} \Delta^{[\pi]}(\bar{w}^{[\pi]}, \hat{y}^{[\pi]}) :=& f^{[\pi]}(\hat{y}^{[\pi]}) - \hat{f}^{[\pi]}_{\bar{w}^{(\pi)}}(\bar{y}^{[\pi]}) \\ =& f^{[\pi]}(\hat{y}^{[\pi]}) - \hat{f}^{[\pi]}_{\bar{w}^{(\pi)}}(\hat{y}^{[\pi]}) + \frac{1}{u} \|g^{[\pi]}(\bar{x}^{[\pi]})\|^2 \\ =& \sum_{r \in R^{[\pi]}} \left[f^{[\pi]}_r(\hat{y}^{[\pi]}) - \hat{f}^{[\pi]}_{\bar{w}^{[\pi]}_r, r}(\hat{y}^{[\pi]}) \right] + \frac{1}{u} \|g^{[\pi]}(\bar{x}^{[\pi]})\|^2. \end{aligned} \tag{5.83}$$

The subspace progress $\Delta^{[\pi]}(\bar{w}^{[\pi]}, \hat{y}^{[\pi]})$ is in direct relation to the expected progress $\Delta(\bar{w}, \hat{y})$ of the global problem, in particular to its part $\Delta_\pi(\bar{w}, \hat{y})$.

Corollary 5.44 *Let π be a subspace problem with subspace $J^{[\pi]} \subseteq M$. Suppose the global data at $\sigma = \underline{\pi}$ is consistent, then*

$$\Delta_\pi(\bar{w}^{\{\underline{\pi}\}}, \hat{y}^{\{\underline{\pi}\}}) = \Delta^{[\pi]}(\bar{w}^{\{\underline{\pi}\}}_{R^{[\pi]}}, \hat{y}^{\{\underline{\pi}\}}_{J^{[\pi]}}).$$

Proof. The proof is similar to the proof of Observation 5.16. Condition (C3) implies that $\bar{x}^{\{\underline{\pi}\}}_r$ satisfies the requirements of Observation 5.43. Thus we get with $\bar{w}^{\{\underline{\pi}\}} = (\bar{l}^{\{\underline{\pi}\}}, \bar{x}^{\{\underline{\pi}\}})$

$$\begin{aligned} \Delta^{[\pi]}(\bar{w}^{\{\underline{\pi}\}}_{R^{[\pi]}}, \hat{y}^{\{\underline{\pi}\}}_{J^{[\pi]}}) &\overset{(5.83)}{=} \sum_{r \in R^{[\pi]}} \left[f^{[\pi]}_r(\hat{y}^{\{\underline{\pi}\}}_{J^{[\pi]}}) - \hat{f}^{[\pi]}_{\bar{w}^{\{\underline{\pi}\}}_r, r}(\hat{y}^{\{\underline{\pi}\}}_{J^{[\pi]}}) \right] + \frac{1}{u} \|g^{[\pi]}(\bar{x}^{[\pi]})\|^2 \\ &= \sum_{r \in R^{[\pi]}} \left[f_r(\hat{y}^{\{\underline{\pi}\}}) - \hat{f}_{\bar{w}^{\{\underline{\pi}\}}_r, r}(\hat{y}^{\{\underline{\pi}\}}) \right] + \frac{1}{u} \|g(\bar{x}^{\{\underline{\pi}\}})_{J^{[\pi]}}\|^2 \\ &\overset{(5.64)}{=} \Delta_\pi(\bar{w}^{\{\underline{\pi}\}}, \hat{y}^{\{\underline{\pi}\}}). \end{aligned}$$

The second equation holds by the previous two observations. □

As in the standard case the subspace problem is not solved to optimality. Instead it is only solved until its solution promises a sufficient improvement for the global problem. In particular, given a parameter $\tau_2 \in (0,1)$, we stop either if the predicted decrease has been sufficiently reduced, *i. e.*

[X1] $$\Delta^{[\pi]}(\bar{w}^{[\pi]}, \hat{y}^{\{\underline{\pi}\}}_{J^{[\pi]}}) < \tau_2 \Delta_\pi(\bar{w}^{\{\underline{\pi}\}}, \hat{y}^{\{\underline{\pi}\}}),$$

or, otherwise, if a descent step occurs, *i. e.*

[X2] $$\Delta^{[\pi]}(\bar{w}^{[\pi]}, \hat{y}^{\{\underline{\pi}\}}_{J^{[\pi]}}) \leq \frac{1}{\varrho}(f^{[\pi]}(\hat{y}^{\{\underline{\pi}\}}_{J^{[\pi]}}) - f^{[\pi]}(\bar{y}^{[\pi]}))$$

where $\bar{y}^{[\pi]} \in \mathbb{R}^{J^{[\pi]}}$ is the current candidate of the bundle algorithm. Note, it is important that these conditions are checked in order, so that

$$\Delta^{[\pi]}(\bar{w}^{[\pi]}, \hat{y}^{\{\underline{\pi}\}}_{J^{[\pi]}}) \geq \tau_2 \Delta_\pi(\bar{w}^{\{\underline{\pi}\}}, \hat{y}^{\{\underline{\pi}\}})$$

holds if a descent step occurs.

The algorithm solving a subspace problem is shown in Algorithm 5.7. This algorithm has no structural difference to Algorithm 5.3 of the standard version except that it uses the new objects.

Algorithm 5.7: Ext-SolveSubspace

Input : process data π
Output : **TRUE** if a descent step occurs
Changes: set final values $\hat{w}^{[\pi]}$, $\bar{w}^{[\pi]}$, $\hat{y}^{[\pi]}$
Starting from initial centre $\hat{y}^{[\pi]} = \hat{y}^{\{\underline{\pi}\}}_{J^{[\pi]}}$ and initial model $\widehat{W}^{[\pi]} = \{\bar{w}^{\{\underline{\pi}\}}_{R^{[\pi]}}\}$,
solve (ExtSub(π)) until either [X1] or [X2] is fulfilled.
This gives

- final centre $\hat{y}^{[\pi]} \in \mathbb{R}^{J^{[\pi]}}$,
- optimal minorant in final centre $\hat{w}^{[\pi]} = (\hat{l}^{[\pi]}, \hat{x}^{[\pi]}) \in W^{[\pi]}$, *i. e.*, for all $r \in R^{[\pi]}$
$$\hat{x}^{[\pi]}_r \in \operatorname{Argmax}\{L^{[\pi]}_r(x_r, \hat{y}^{[\pi]}) \colon x_r \in \mathcal{X}_r\},$$
- $\bar{w}^{[\pi]} \in \operatorname{conv} W^{[\pi]}$.

if *[X1] is fulfilled* **then**
  // $\hat{y}^{[\pi]} = \hat{y}^{\{\underline{\pi}\}}_{J^{[\pi]}}$ is the old centre, $\hat{w}^{[\pi]} = \hat{w}^{\{\underline{\pi}\}}_{R^{[\pi]}}$ an optimal minorant in the old centre, $\bar{w}^{[\pi]}$ the new aggregate minorant
  return *FALSE*
else
  // [X2] is fulfilled
  // $\hat{y}^{[\pi]}$ is the new centre, $\hat{w}^{[\pi]}$ an optimal minorant in the new centre, $\bar{w}^{[\pi]}$ the new aggregate minorant
  return *TRUE*

Because Algorithm 5.7 runs a bundle method it finishes in finite time.

Observation 5.45 *(see Observation 5.17)*
Algorithm 5.7 is finite.

Proof. Analogous to the proof of Observation 5.17. □

The termination conditions ensure again that in case [X2] the descent on the subspace is sufficiently large compared with the predicted global descent on the whole space when π starts.

Observation 5.46 *(see Observation 5.18)*
Let π be a subspace problem with subspace $J \subseteq M$, and assume that Algorithm 5.7 stopped because of [X2] with final values $\bar{w}^{[\pi]} \in \operatorname{conv} W^{[\pi]}$ and $\bar{y}^{[\pi]} \in \mathbb{R}^{J^{[\pi]}}$. Then

$$f^{[\pi]}(\hat{y}^{\{\pi\}}_{J^{[\pi]}}) - f^{[\pi]}(\bar{y}^{[\pi]}) \geq \tau_2 \varrho \Delta_\pi(\bar{w}^{\{\pi\}}, \hat{y}^{\{\pi\}}).$$

Proof. Since the algorithm stopped because of [X2] and *not* because of [X1] we know

$$\Delta^{[\pi]}(\bar{w}^{[\pi]}, \hat{y}^{\{\pi\}}_{J^{[\pi]}}) \geq \tau_2 \Delta_\pi(\bar{w}^{\{\pi\}}, \hat{y}^{\{\pi\}})$$

and

$$\Delta^{[\pi]}(\bar{w}^{[\pi]}, \hat{y}^{\{\pi\}}_{J^{[\pi]}}) \leq \frac{1}{\varrho}(f^{[\pi]}(\hat{y}^{\{\pi\}}_{J^{[\pi]}}) - f^{[\pi]}(\bar{y}^{[\pi]})).$$

The result follows by putting these two inequalities together. □

5.4.6 Update of the Global Data

The last step of the algorithm contains the most significant changes w. r. t. the standard version. As before, this step is executed with exclusive access to the global data when a process has finished the processing of its subspace problem. Depending on whether the subspace problem stopped with a descent step or with sufficiently decreased predicted progress, the update is slightly different. In the latter case an update of the dependency graph may be required. In addition, and in contrast to the standard case, also the activity sets $J_r^{\{\sigma\}}$, $r \in R$, must be updated, no matter which condition causes the solution process to stop.

The parallelism of the algorithm adds another complicating aspect. When the subprocess started, the subspace selection and the construction of the subspace problem were made under the assumptions of certain dependencies between subproblems and constraints, namely the activity sets $J_r^{\{\pi\}}$, $r \in R$. These activity sets may not only change because of the subprocess itself, but they may also be changed by other processes while π is running.

These changes in the activity sets may cause the newly computed values to be wrong. In this case the computed solutions must not be written to the global data but must be discarded.

In more detail, if the algorithm cannot guarantee that the updated global data at $J_r^{\{\bar{\pi}+1\}}$ is still consistent, the computed values will be dropped. Let $J_r^{\{\bar{\pi}+1\}}$ be the new activity sets (they will be computed in the update step). Before accepting the computed values $\hat{y}_j^{[\pi]}$, $j \in J^{[\pi]}$, and $\hat{x}_r^{[\pi]}$, $\bar{x}_r^{[\pi]}$, $r \in R^{[\pi]}$, as new global values, the algorithm tests if the following conditions hold

$$\hat{y}_j^{[\pi]} C_{j,r} x_r \geq 0, \qquad \text{for all } r \in R^{[\pi]}, x_r \in \mathcal{X}_r, j \in J^{[\pi]} \cap (J_r^{\{\bar{\pi}+1\}} \setminus J_r^{\{\underline{\pi}\}}), \tag{T1}$$

$$\hat{y}_j^{[\pi]} C_{j,r} \hat{x}_r^{[\pi]} = 0, \qquad \text{for all } r \in R^{[\pi]}, j \in (J_r^{\{\bar{\pi}+1\}} \setminus J_r^{\{\underline{\pi}\}}) \cap J^{[\pi]}, \tag{T2.1}$$

$$\hat{y}_j^{\{\bar{\pi}\}} C_{j,r} \hat{x}_r^{[\pi]} = 0, \qquad \text{for all } r \in R^{[\pi]}, j \in (J_r^{\{\bar{\pi}+1\}} \setminus J_r^{\{\underline{\pi}\}}) \setminus J^{[\pi]}, \tag{T2.2}$$

$$\hat{y}_j^{[\pi]} C_{j,r} \bar{x}_r^{[\pi]} = 0, \qquad \text{for all } r \in R^{[\pi]}, j \in (J_r^{\{\bar{\pi}+1\}} \setminus J_r^{\{\underline{\pi}\}}) \cap J^{[\pi]}, \tag{T3.1}$$

$$\hat{y}_j^{\{\bar{\pi}\}} C_{j,r} \bar{x}_r^{[\pi]} = 0, \qquad \text{for all } r \in R^{[\pi]}, j \in (J_r^{\{\bar{\pi}+1\}} \setminus J_r^{\{\underline{\pi}\}}) \setminus J^{[\pi]}, \tag{T3.2}$$

$$J_r^{\{\bar{\pi}+1\}} \cap J^{[\pi]} = \emptyset, \qquad \text{for all } r \in R_{J^{[\pi]}} \setminus R^{[\pi]}. \tag{T4}$$

If these conditions hold, then the new data at $\sigma + 1$, which will be set to

$$\begin{aligned}
(\hat{y}_{J^{[\pi]}}^{\{\bar{\pi}+1\}}, \hat{y}_{M \setminus J^{[\pi]}}^{\{\bar{\pi}+1\}}) &= (\hat{y}^{[\pi]}, \hat{y}_{M \setminus J^{[\pi]}}^{\{\bar{\pi}\}}), \\
(\hat{x}_{R^{[\pi]}}^{\{\bar{\pi}+1\}}, \hat{x}_{R \setminus R^{[\pi]}}^{\{\bar{\pi}+1\}}) &= (\hat{x}^{[\pi]}, \hat{x}_{R \setminus R^{[\pi]}}^{\{\bar{\pi}\}}), \\
(\bar{x}_{R^{[\pi]}}^{\{\bar{\pi}+1\}}, \bar{x}_{R \setminus R^{[\pi]}}^{\{\bar{\pi}+1\}}) &= (\bar{x}^{[\pi]}, \bar{x}_{R \setminus R^{[\pi]}}^{\{\bar{\pi}\}}),
\end{aligned}$$

will be consistent and be accepted as the new global values. The proof of this is very technical and is postponed to the next section.

The update algorithm is shown in Algorithm 5.8.

Remark 5.47 Note that the consistency tests A5.8.3 check constraints $j \in J_r^{\{\bar{\pi}+1\}} \setminus J_r^{\{\underline{\pi}\}}$, not only $j \in J_r^{\{\bar{\pi}+1\}} \setminus J_r^{\{\bar{\pi}\}}$. In particular it may hold $J_r^{\{\bar{\pi}+1\}} = J_r^{\{\bar{\pi}\}} \supsetneq J_r^{\{\underline{\pi}\}}$ if some other process increased the activity set of r in A5.8.2 between $\underline{\pi}$ and $\bar{\pi}$. Therefore it may happen that the consistency tests fail even if π does not increase any activity set, *i. e.*, $J_r^{\{\bar{\pi}\}} = J_r^{\{\bar{\pi}+1\}}$ for all $r \in R$. The necessity of these test will become apparent later (see Lemma 5.49).

5.4.7 The Extended Parallel Bundle Algorithm

Putting the three steps together we can form the extended parallel bundle algorithm. It is given as Algorithm 5.9.

Algorithm 5.8: EXT-UPDATESUBSPACE

Input : process data π, $\mathit{Descent} \in \{TRUE, FALSE\}$
Changes: sets global data at $\bar{\pi}+1$

`// Update activity sets`

$J_{R\setminus R^{[\pi]}}^{\{\bar{\pi}+1\}} \leftarrow J_{R\setminus R^{[\pi]}}^{\{\bar{\pi}\}}$

for $r \in R^{[\pi]}$ **do**

1 $\quad J_r^{\{\bar{\pi}+1\}} \leftarrow J_r^{\{\bar{\pi}\}} \cup \{j \in J_r \setminus J_r^{\{\bar{\pi}\}} : C_{j,r}\hat{x}_r^{[\pi]} \neq 0 \vee C_{j,r}\bar{x}_r^{[\pi]} \neq 0\}$
$\qquad\qquad\qquad\qquad \cup \{j \in J^{[\pi]} \cap J_r \setminus J_r^{\{\bar{\pi}\}} : \hat{y}^{[\pi]} \cdot C_{j,r}x_r < 0 \text{ for some } x_r \in \mathcal{X}_r\}$

for $r \in R_{J^{[\pi]}} \setminus R^{[\pi]}$ **do**

2 $\quad J_r^{\{\bar{\pi}+1\}} \leftarrow J_r^{\{\bar{\pi}\}} \cup \{j \in J^{[\pi]} \cap J_r \setminus J_r^{\{\bar{\pi}\}} : \hat{y}_j^{[\pi]} \cdot C_{j,r}x_r < 0 \text{ for some } x_r \in \mathcal{X}_r\}$

`// Ensure consistency of the new solution`

3 **if** $\hat{x}^{[\pi]}, \bar{x}^{[\pi]}, \hat{y}^{[\pi]}$ *fulfil (T1)–(T4)* **then**

$\quad$ `// Accept new solution`

4 $\quad (\hat{y}_{J^{[\pi]}}^{\{\bar{\pi}+1\}}, \hat{y}_{M\setminus J^{[\pi]}}^{\{\bar{\pi}+1\}}) \leftarrow (\hat{y}^{[\pi]}, \hat{y}_{M\setminus J^{[\pi]}}^{\{\bar{\pi}\}})$ `// Update centre of stability on` $J^{[\pi]}$

5 $\quad (\hat{w}_{R^{[\pi]}}^{\{\bar{\pi}+1\}}, \hat{w}_{R\setminus R^{[\pi]}}^{\{\bar{\pi}+1\}}) \leftarrow (\hat{w}^{[\pi]}, \hat{w}_{R\setminus R^{[\pi]}}^{\{\bar{\pi}\}})$ `// Update minorant in centre on` $R^{[\pi]}$

6 $\quad (\bar{w}_{R^{[\pi]}}^{\{\bar{\pi}+1\}}, \bar{w}_{R\setminus R^{[\pi]}}^{\{\bar{\pi}+1\}}) \leftarrow (\bar{w}^{[\pi]}, \bar{w}_{R\setminus R^{[\pi]}}^{\{\bar{\pi}\}})$ `// Update aggregate minorant on` $R^{[\pi]}$

7 $\quad$ **if** $\mathit{Descent} = \boldsymbol{FALSE}$ *and* $\delta_\pi(\bar{w}^{\{\bar{\pi}+1\}}) - \delta_\pi(\bar{w}^{\{\bar{\pi}\}}) > \tau_3\Delta_\pi(\bar{w}^{\{\underline{\pi}\}}, \hat{y}^{\{\underline{\pi}\}})$ **then**

$\quad\quad$ `// Update dependency graph with` $\bar{w}^{\{\bar{\pi}\}} = (\bar{l}^{\{\bar{\pi}\}}, \bar{x}^{\{\bar{\pi}\}})$ `and` $\bar{w}^{\{\bar{\pi}+1\}} = (\bar{l}^{\{\bar{\pi}+1\}}, \bar{x}^{\{\bar{\pi}+1\}})$

$\quad\quad$ choose $(j, j') \in \operatorname{Argmax}\left\{g(\bar{x}^{\{\bar{\pi}+1\}})_{\tilde{j}}^2 - g(\bar{x}^{\{\bar{\pi}\}})_{\tilde{j}}^2 : (\tilde{j}, \hat{j}) \in (J^{[\pi]} \times \bar{J}^{[\pi]}) \setminus E^{\{\bar{\pi}\}}\right\}$

8 $\quad\quad E^{\{\bar{\pi}+1\}} \leftarrow E^{\{\bar{\pi}\}} \cup \{(j, j')\}$

$\quad$ **else**

9 $\quad\quad E^{\{\bar{\pi}+1\}} \leftarrow E^{\{\bar{\pi}\}}$

else

$\quad$ `// Discard solution, keep old values`

10 $\quad (\hat{y}^{\{\bar{\pi}+1\}}, \hat{w}^{\{\bar{\pi}+1\}}, \bar{w}^{\{\bar{\pi}+1\}}) \leftarrow (\hat{y}^{\{\bar{\pi}\}}, \hat{w}^{\{\bar{\pi}\}}, \bar{w}^{\{\bar{\pi}\}})$

Remark 5.48 Note that the values $\hat{x}_r^{\{\sigma\}}$ with $\hat{w}_r^{\{\sigma\}} = (\hat{l}_r^{\{\sigma\}}, \hat{x}_r^{\{\sigma\}})$, $r \in R$, are only used in Algorithm 5.8. In fact, needs only the values $\hat{x}_r^{[\pi]}$, $r \in R^{[\pi]}$, computed in the current process. Thus there is no need to store the global values $\hat{x}_r^{\{\sigma\}}$ at all in a practical implementation. However, we will keep them in the algorithm to simplify the theoretic considerations and proofs in the next sections.

5.4.8 Consistency of the Update Scheme

In this section we analyse the relations between the local and global data in each subprocess. The analysis follows along the same lines as in the standard case. But this time we have to observe the (changing) activity sets $J_r^{\{\sigma\}}$, $r \in R$. One of the most critical aspects, that differs from the standard case, is that we have to show that the global data is consistent for each σ. It will turn out that we must carefully analyse each possible outcome of the subspace problem (null-step, descent-step or discarding of solutions). This leads to very technical results in this section.

The basic idea is quite easy. As in the standard case we prove inductively that the global data used to set up a subspace problem and that is updated by a subprocess matches in some sense with the local data computed by the subprocesses. In addition we must show that the data at $\sigma + 1$ is *consistent*. For the components of $\hat{x}^{\{\sigma\}}$, $\bar{x}^{\{\sigma\}}$ and $\hat{y}^{\{\sigma\}}$, that are not modified by a process π, this will follow by induction. For components that *are* modified by a process π, *e. g.* $\hat{x}_{R^{[\pi]}}^{\{\sigma\}}$, $\bar{x}_{R^{[\pi]}}^{\{\sigma\}}$ and $\hat{y}_{J^{[\pi]}}^{\{\sigma\}}$, this validity will be ensured by tests (T1)–(T4). In other words, the idea is to enforce the validity of all those objects that are not valid automatically (by induction) by explicit tests .

We start with the investigation of an update step with a successful consistency test.

Lemma 5.49 *Assume process π satisfies the consistency check A5.8.3 at $\bar{\pi}$ and let $r \in R^{[\pi]}$. Suppose $(\hat{y}^{\{\bar{\pi}\}}, \hat{x}_r^{\{\bar{\pi}\}}, \bar{x}_r^{\{\bar{\pi}\}}, J_r^{\{\underline{\pi}\}})$ is consistent and $\hat{y}_{\bar{J}^{[\pi]}}^{\{\bar{\pi}\}} = \hat{y}_{\bar{J}^{[\pi]}}^{\{\underline{\pi}\}}$. Let $\bar{x}_r^{[\pi]}$ denote the final aggregated primal and $\hat{x}_r^{[\pi]}$ the final optimal solution of the subspace problem in the new centre $\hat{y}_{J^{[\pi]}}^{\{\bar{\pi}+1\}}$, i. e.*

$$\hat{x}_r^{[\pi]} \in \operatorname{Argmax}\left\{ L_r^{[\pi]}(x_r, \hat{y}_{J^{[\pi]}}^{\{\bar{\pi}+1\}}) \colon x_r \in \mathcal{X}_r \right\}.$$

Then $(\hat{y}^{\{\bar{\pi}+1\}}, \hat{x}_r^{[\pi]}, \bar{x}_r^{[\pi]}, J_r^{\{\underline{\pi}\}})$ is weakly consistent and $(\hat{y}^{\{\bar{\pi}+1\}}, \hat{x}_r^{[\pi]}, \bar{x}_r^{[\pi]}, J_r^{\{\bar{\pi}+1\}})$ is consistent for r.

Proof. First observe that $\hat{y}^{\{\bar{\pi}+1\}} = (\hat{y}^{[\pi]}, \hat{y}_{J \setminus J^{[\pi]}}^{\{\bar{\pi}\}})$ by A5.8.4. In order to prove (C1), we have to show that $\hat{y}_j^{\{\bar{\pi}+1\}} C_{j,r} x_r \geq 0$ for all $x_r \in \mathcal{X}_r$ and all $j \in J_r \setminus J_r^{\{\underline{\pi}\}}$.

Algorithm 5.9: Ext-ParallelBundle

Input : Problem (P), parameters
- $\tau_1 \in (0, \frac{1}{2}\min\{\frac{1}{|M|}, \frac{1}{|R|}\})$, $\tau_2 \in (0,1)$, $\tau_3 \in [0, 1-\tau_2)$, $\varepsilon > 0$, $u > 0$
- $N_\Pi \in \{1, \dots, |M|\}$

Output: Approximate solutions $x \in \operatorname{conv} \mathcal{X}$, $y \in \mathbb{R}^M$

set $\sigma \leftarrow 0$, $\hat{y}^{\{0\}} \leftarrow 0$, $B^{\{0\}} \leftarrow \emptyset$, $B_M^{\{0\}} \leftarrow \emptyset$

set $E^{\{0\}} \leftarrow \emptyset$ `// or use some prespecified dependencies`

set $\hat{w}^{\{0\}} = (\hat{l}^{\{0\}}, \hat{x}^{\{0\}})$ with $\hat{x}_r^{\{0\}} \in \operatorname{Argmax}\{L_r^{[\pi]}(x_r, \hat{y}^{\{0\}}) \colon x_r \in \mathcal{X}_r\}$ and

$$\hat{l}_r^{\{0\}} = L_r^{[\pi]}(\hat{x}^{\{0\}}, \hat{y}^{\{0\}}) = f_r(\hat{y}^{\{0\}}), r \in R$$

set $\bar{w}^{\{0\}} = \hat{w}^{\{0\}}$ `// the minorant defined by the optimal solutions`

set $J_r^{\{0\}} \leftarrow \{j \in M \colon C_{j,r}\bar{x}_r^{\{0\}} \neq 0\}$, $r \in R$

if $\Delta(\bar{w}^{\{0\}}, \hat{y}^{\{0\}}) \leq \varepsilon(|f(\hat{y}^{\{0\}})| + 1)$ **then**
- then do not start any process
- set $x \leftarrow \bar{x}^{\{0\}}$, $y \leftarrow \hat{y}^{\{0\}}$
- **STOP**

while *Less than* N_Π *processes are running* **do**
- Start a new process $\pi = (\underline{\pi}, \bar{\pi}) \leftarrow (-1, -1)$.
- **Each** *process* π *performs the following steps independently*
  - **Secure exclusive access to global data**
  - $\underline{\pi} \leftarrow \sigma$
  - **if** Ext-SelectSubspace$(\underline{\pi})$ = ***FALSE*** **then**
    - `// Step is unsuccessful`
    - **Free exclusive access to global data**
    - **STOP** this process
  - 3 $(\hat{y}^{\{\underline{\pi}+1\}}, \hat{w}^{\{\underline{\pi}+1\}}, \bar{w}^{\{\underline{\pi}+1\}}, E^{\{\underline{\pi}+1\}}, J_R^{\{\underline{\pi}+1\}}) \leftarrow (\hat{y}^{\{\underline{\pi}\}}, \hat{w}^{\{\underline{\pi}\}}, \bar{w}^{\{\underline{\pi}\}}, E^{\{\underline{\pi}\}}, J_R^{\{\underline{\pi}\}})$
  - 4 $B^{\{\underline{\pi}+1\}} \leftarrow B^{\{\underline{\pi}\}} \cup R^{[\pi]}$, $B_M^{\{\underline{\pi}+1\}} \leftarrow B_M^{\{\underline{\pi}\}} \cup J^{[\pi]}$
  - $\bar{\pi} \leftarrow \infty$, $\sigma \leftarrow \underline{\pi} + 1$
  - **Free exclusive access to global data**
  - `// Solve the subspace problem to sufficient precision`
  - 5 *Descent* $\leftarrow$ Ext-SolveSubspace(π)
  - **Secure exclusive access to global data**
  - $\bar{\pi} \leftarrow \sigma$
  - 6 Ext-UpdateSubspace$(\pi,$ *Descent*$)$
  - 7 $B^{\{\bar{\pi}+1\}} \leftarrow B^{\{\bar{\pi}\}} \setminus R^{[\pi]}$, $B_M^{\{\bar{\pi}+1\}} \leftarrow B_M^{\{\bar{\pi}\}} \setminus J^{[\pi]}$
  - $\sigma \leftarrow \bar{\pi} + 1$
  - 8 **if** $\Delta(\bar{w}^{\{\bar{\pi}+1\}}, \hat{y}^{\{\bar{\pi}+1\}}) \leq \varepsilon(|f(\hat{y}^{\{\bar{\pi}+1\}})| + 1)$ **then**
    - set $x \leftarrow \bar{x}^{\{\sigma\}}$, $y \leftarrow \hat{y}^{\{\sigma\}}$
    - **TERMINATE** *all* processes and **STOP**.
  - **Free exclusive access to global data**
  - **STOP** this process π.

- If $j \in (J_r \setminus J_r^{\{\underline{\pi}\}}) \setminus J^{[\pi]}$ then $\hat{y}_j^{\{\bar{\pi}+1\}} = \hat{y}_j^{\{\bar{\pi}\}}$, thus the assertion follows from the consistency of $(\hat{y}^{\{\bar{\pi}\}}, \hat{x}_r^{\{\bar{\pi}\}}, \bar{x}_r^{\{\bar{\pi}\}}, J_r^{\{\underline{\pi}\}})$.
- If $j \in (J_r \setminus J_r^{\{\bar{\pi}+1\}}) \cap J^{[\pi]}$ then the update of the activity set in A5.8.1 implies the assertion.
- If $j \in (J_r^{\{\bar{\pi}+1\}} \setminus J_r^{\{\underline{\pi}\}}) \cap J^{[\pi]}$ then the consistency test (T1) implies the assertion.

Putting all together (C1) holds.

Next we prove that (C2'), (C3') for $J_r^{\{\underline{\pi}\}}$ and (C2), (C3) for $J_r^{\{\bar{\pi}+1\}}$ hold. Observe that the update of the activity set in A5.8.1 includes all indexes $j \in J_r \setminus J_r^{\{\bar{\pi}\}}$ with $C_{j,r}\bar{x}_r^{[\pi]} \neq 0$ or $C_{j,r}\hat{x}_r^{[\pi]} \neq 0$, so $C_{j,r}\bar{x}_r^{[\pi]} = 0 = C_{j,r}\hat{x}_r^{[\pi]}$ for all $j \in J_r \setminus J_r^{\{\bar{\pi}+1\}}$, proving (C2), (C3) for $J_r^{\{\bar{\pi}+1\}}$. Furthermore, the successful consistency tests (T2.1)–(T3.2) certify that $\hat{y}_j^{\{\bar{\pi}+1\}} C_{j,r}\bar{x}_r^{[\pi]} = 0 = \hat{y}_j^{\{\bar{\pi}+1\}} C_{j,r}\hat{x}_r^{[\pi]}$ for all $j \in J_r^{\{\bar{\pi}+1\}} \setminus J_r^{\{\underline{\pi}\}}$. Consequently (C2') and (C3') hold for $J_r^{\{\underline{\pi}\}}$.

Finally we show (C4) for $\hat{x}_r^{[\pi]}$. Because $\hat{y}_{\bar{J}^{[\pi]}}^{\{\bar{\pi}+1\}} = \hat{y}_{\bar{J}^{[\pi]}}^{\{\bar{\pi}\}} = \hat{y}_{\bar{J}^{[\pi]}}^{\{\underline{\pi}\}}$ by assumption, we can apply Observation 5.41. So by its choice $\hat{x}_r^{[\pi]}$ fulfils

$$\begin{aligned} \hat{x}_r^{[\pi]} &\in \operatorname{Argmax}\{L_r^{[\pi]}(x_r, \hat{y}_{J^{[\pi]}}^{\{\bar{\pi}+1\}}) \colon x_r \in \mathcal{X}_r\} \\ &\overset{(5.77)}{=} \operatorname{Argmax}\{h_r(x_r) - (\hat{y}_{J_r^{\{\underline{\pi}\}}}^{\{\bar{\pi}+1\}})^T C_{J_r^{\{\underline{\pi}\}},r} x_r \colon x_r \in \mathcal{X}_r\}, \end{aligned}$$

i. e., (C4) holds. This completes the proof. □

The next result collects a number of relations that hold throughout the algorithm. It is a generalisation and extension of Lemma 5.20 to the extended algorithm.

Lemma 5.50 *For $\sigma \in \Sigma$,*

(i) $B^{\{\sigma\}} = \bigcup_{\pi \in \Pi^{\{\sigma\}}} R^{[\pi]}, B_M^{\{\sigma\}} = \bigcup_{\pi \in \Pi^{\{\sigma\}}} J^{[\pi]}$,

(ii) $R^{[\pi]} \cap R^{[\pi']} = \emptyset$ *for* $\pi, \pi' \in \Pi^{\{\sigma\}}$ *with* $\pi \neq \pi'$,

(iii) $J^{[\pi']} \cap (J^{[\pi]} \cup \bar{J}^{[\pi]}) = \emptyset = J^{[\pi]} \cap (J^{[\pi']} \cup \bar{J}^{[\pi']})$ *for* $\pi, \pi' \in \Pi^{\{\sigma\}}$ *with* $\pi \neq \pi'$,

(iv) $\hat{y}_{J^{[\pi]} \cup \bar{J}^{[\pi]}}^{\{\underline{\pi}\}} = \hat{y}_{J^{[\pi]} \cup \bar{J}^{[\pi]}}^{\{\sigma\}}$, $\hat{w}_{R^{[\pi]}}^{\{\underline{\pi}\}} = \hat{w}_{R^{[\pi]}}^{\{\sigma\}}$, *and* $\bar{w}_{R^{[\pi]}}^{\{\underline{\pi}\}} = \bar{w}_{R^{[\pi]}}^{\{\sigma\}}$, *for* $\pi \in \Pi^{\{\sigma\}}$,

(v) $J_r^{\{\sigma'\}} \subseteq J_r^{\{\sigma\}} \subseteq J_r$ *for* $\sigma' \in \Sigma$ *with* $\sigma' < \sigma$ *and* $r \in R$,

(vi) $J_r^{[\pi]} = J_r^{\{\underline{\pi}\}} \cap J^{[\pi]} = J_r^{\{\sigma\}} \cap J^{[\pi]}$ *for* $r \in R^{[\pi]}$, $\pi \in \Pi^{\{\sigma\}}$,

(vii) the global data at σ is consistent,

(viii) $(\hat{y}^{\{\sigma\}}, \hat{x}_r^{\{\sigma\}}, \bar{x}_r^{\{\sigma\}}, J_r^{\{\underline{\pi}\}})$ *is consistent for* $r \in R^{[\pi]}$, $\pi \in \Pi^{\{\sigma\}}$.

Proof. The proof works by induction on σ. For $\sigma = 0$ the initialisation step A5.9.1 sets $B^{\{0\}} = \emptyset = B_M^{\{0\}}$, $\hat{y}^{\{0\}} = 0$, $\hat{w}^{\{0\}} = \bar{w}^{\{0\}} = (\bar{l}^{\{0\}}, \bar{x}^{\{0\}})$ and the activity sets $J_r^{\{0\}}$ $(r \in R)$ so that $C_{j,r}\bar{x}_r^{\{0\}} \neq 0 \Rightarrow j \in J_r^{\{0\}}$ for $r \in R$. Hence the global data at $\sigma = 0$ is consistent and together with $\Pi^{\{0\}} = \emptyset$ relations (i)–(viii) hold.

Now suppose $\sigma + 1 \in \Sigma$ and the claim holds for $\sigma \in \Sigma$. By definition of Σ, $\sigma + 1 \in \Sigma$ implies $\Pi^{\{\sigma\}} \neq \Pi^{\{\sigma+1\}}$, so Observation 5.10 asserts the existence of a unique $\eta \in (\Pi^{\{\sigma\}} \setminus \Pi^{\{\sigma+1\}}) \cup (\Pi^{\{\sigma+1\}} \setminus \Pi^{\{\sigma\}})$ and this η either satisfies $\underline{\eta} = \sigma$ or $\bar{\eta} = \sigma$.

Case $\underline{\eta} = \sigma$:
We have $\Pi^{\{\sigma\}} = \Pi^{\{\sigma+1\}} \setminus \{\eta\}$ and process η executes a successful subspace selection step at σ. In this case the update A5.9.3 implies that most of the data remains unchanged (in fact, only the blocking sets $B^{\{\sigma\}}$ and $B_M^{\{\sigma\}}$ are changed in A5.9.4).

(i) and (iv)–(vii): These hold because in a successful subspace selection step

$$B^{\{\sigma+1\}} = B^{\{\sigma\}} \cup R^{[\pi]}$$

and

$$B_M^{\{\sigma+1\}} = B_M^{\{\sigma\}} \cup J^{[\pi]}$$

are the only global objects that are changed.

(ii) and (iii): By induction, these only need to be verified for $\pi = \eta$ and $\pi' \in \Pi^{\{\sigma+1\}} \setminus \{\eta\} = \Pi^{\{\sigma\}}$. We know $R^{[\pi']} \subseteq B^{\{\sigma\}}$ by (i) and Observation 5.40 shows $R^{[\eta]} \cap B^{\{\sigma\}} = \emptyset$. So (ii) follows from

$$\emptyset = R^{[\eta]} \cap B^{\{\sigma\}} \supseteq R^{[\eta]} \cap R^{[\pi']}.$$

Applying the corresponding arguments to $J^{[\pi]}$ and $B_M^{\{\sigma\}}$ proves the left-hand side equation of (iii). In particular $J^{[\pi']} \cap J^{[\eta]} = \emptyset$.

For the right-hand side equation of (iii) assume there exists a $j \in J^{[\eta]} \cap (J^{[\pi']} \cup \bar{J}^{[\pi']})$, then by the left-hand side equation we have $j \in \bar{J}^{[\pi']}$. The definition (5.58) of $\bar{J}^{[\pi']}$ implies that there is an $r \in R^{[\pi']}$ with $j \in J_r^{\{\underline{\pi}'\}}$. Now $\underline{\pi}' < \underline{\eta} = \sigma$ by the choice of η, so (v) implies $J_r^{\{\underline{\pi}'\}} \subseteq J_r^{\{\underline{\eta}\}}$ and with $j \in J^{[\eta]}$ we conclude

$$r \in R^{[\eta]} \overset{(5.55)}{=} \bigcup_{j \in J^{[\eta]}} \{r \in R \colon j \in J_r^{\{\underline{\eta}\}}\},$$

contradicting (ii).

(viii): Because the data involved in the conditions is not modified, they hold by induction for $\pi' \in \Pi^{\{\sigma+1\}} \setminus \{\eta\}$. So it remains to consider η. Fix some $r \in R^{[\eta]}$. Then because

$$J_r^{\{\underline{\eta}\}} \overset{A5.9.3}{=} J_r^{\{\underline{\eta}+1\}} \overset{\sigma=\underline{\eta}}{=} J_r^{\{\sigma+1\}},$$

(viii) is implied by (vii) for $\sigma + 1$.

Case $\bar{\eta} = \sigma$:
We have $\Pi^{\{\sigma+1\}} = \Pi^{\{\sigma\}} \setminus \{\eta\}$ and process η executed an update step at σ, in particular called Algorithm 5.8 in A5.9.6. This time the validity of many relations will be guaranteed by the tests within Algorithm 5.8.

(i)–(iv): The update of the blocking sets in A5.9.7 gives $B^{\{\sigma+1\}} = B^{\{\sigma\}} \setminus R^{[\eta]}$, $B_M^{\{\sigma+1\}} = B_M^{\{\sigma\}} \setminus J^{[\eta]}$, which, together with (ii) for σ, proves (i). Because $\Pi^{\{\sigma+1\}} = \Pi^{\{\sigma\}} \setminus \{\eta\}$, (ii) and (iii) follow by induction for all processes $\pi, \pi' \in \Pi^{\{\sigma+1\}}$.

Now fix $\pi \in \Pi^{\{\sigma+1\}}$. If process η modifies the global values $\hat{y}^{\{\sigma\}}$, $\hat{w}^{\{\sigma\}}$ and $\bar{w}^{\{\sigma\}}$ then only on the indexes $J^{[\eta]}$ and $R^{[\eta]}$, respectively, due to lines A5.8.4 – A5.8.6. Thus (ii) and (iii) for $\pi \neq \eta \in \Pi^{\{\sigma\}}$ imply that the relevant indexes for (iv) remain unchanged.

(v): Its correctness follows directly from the operations performed on $J_r^{\{\sigma+1\}}$ for $r \in R$ in A5.8.1 and A5.8.2 (the activity sets are only increased).

(vi): Because of (v) and the induction hypothesis it suffices to show

$$J_r^{\{\sigma+1\}} \cap J^{[\pi]} \subseteq J_r^{\{\pi\}} \cap J^{[\pi]} = J_r^{\{\sigma\}} \cap J^{[\pi]}$$

for $r \in R^{[\pi]}$, $\pi \in \Pi^{\{\sigma+1\}}$. Assume, for contradiction, for some $\pi' \in \Pi^{\{\sigma+1\}} = \Pi^{\{\sigma\}} \setminus \{\eta\}$ there is an $r \in R^{[\pi']}$ with some $j \in (J_r^{\{\sigma+1\}} \cap J^{[\pi']} \setminus J_r^{\{\sigma\}})$. Now $r \notin R^{[\eta]}$ by (ii) for $\pi = \eta$ at σ. Furthermore, because this j is contained in $J_r^{\{\sigma+1\}}$ but not in $J_r^{\{\sigma\}}$ it was added to the activity set in line A5.8.2 in the update step of η at σ. But this forces $j \in J^{[\eta]}$. This contradicts (iii) for $\pi = \eta$ at σ.

(vii): We first consider the case that the consistency tests in A5.8.3 fail. In this case the new solution is discarded in A5.8.10 and we have $\hat{x}_r^{\{\sigma+1\}} = \hat{x}_r^{\{\sigma\}}$, $\bar{x}_r^{\{\sigma+1\}} = \bar{x}_r^{\{\sigma\}}$, $r \in R$, and $\hat{y}^{\{\sigma+1\}} = \hat{y}^{\{\sigma\}}$. Together with (v) we can apply Observation 5.38 and (vii) holds for $\sigma + 1$. So we may assume in the following that the consistency tests succeed and there is a "real" update of the global data at $\sigma = \bar{\eta}$.

First let $r \in R^{[\eta]}$. By induction hypothesis for (viii) we know $(\hat{y}^{\{\sigma\}}, \hat{x}_r^{\{\sigma\}}, \bar{x}_r^{\{\sigma\}}, J_r^{\{\eta\}})$ is consistent and by (iv) we have $\hat{y}^{\{\sigma\}}_{\bar{J}^{[\eta]}} = \hat{y}^{\{\pi\}}_{\bar{J}^{[\eta]}}$. Thus we can apply Lemma 5.49 and together with the updates A5.8.5 and A5.8.6 we conclude the consistency of $(\hat{y}^{\{\sigma+1\}}, \hat{x}_r^{\{\sigma+1\}}, \bar{x}_r^{\{\sigma+1\}}, J_r^{\{\sigma+1\}})$.

Now let $r \in R \setminus R^{[\eta]}$. We show

$$J_r^{\{\sigma\}} = J_r^{\{\sigma+1\}} \quad \text{and} \quad J_r^{\{\sigma+1\}} \cap J^{[\eta]} = \emptyset.$$

If $r \in R \setminus R_{J^{[\eta]}}$ then the first equation holds because neither A5.8.1 nor A5.8.2 adds a constraint to the activity set, and the second equation holds because of

$$J_r^{\{\sigma+1\}} \cap J^{[\eta]} \overset{(v)}{\subseteq} J_r \cap J^{[\eta]} \overset{r \notin R_{J^{[\eta]}}}{=} \emptyset.$$

If $r \in R_{J^{[\eta]}} \setminus R^{[\eta]}$, then both equations hold because of the update in A5.8.2 and the subsequent consistency test (T4).

Consequently, the updates in A5.8.4 – A5.8.6 imply $\hat{y}^{\{\sigma\}}_{J_r^{\{\sigma\}}} = \hat{y}^{\{\sigma+1\}}_{J_r^{\{\sigma+1\}}}$, $\hat{x}_r^{\{\sigma\}} = \hat{x}_r^{\{\sigma+1\}}$ and $\bar{x}_r^{\{\sigma\}} = \bar{x}_r^{\{\sigma+1\}}$. Hence, (C2)–(C4) hold by induction because none of the involved data changed. Furthermore, the update A5.8.2 implies for $j \in (J_r \setminus J_r^{\{\sigma+1\}}) \cap J^{[\eta]}$

$$\hat{y}_j^{\{\sigma+1\}} C_{j,r} x_r \geq 0, \quad \text{for all } x_r \in \mathcal{X}_r,$$

so (C1) holds as well. This completes the proof of (vii).

(viii): Fix $\pi' \in \Pi^{\{\sigma+1\}}$ and $r \in R^{[\pi']}$. By induction $(\hat{y}^{\{\sigma\}}, \hat{x}_r^{\{\sigma\}}, \bar{x}_r^{\{\sigma\}}, J_r^{\{\underline{\pi}'\}})$ is consistent. In particular, $\hat{x}_r^{\{\sigma\}}$, $\bar{x}_r^{\{\sigma\}}$ satisfy (C2) resp. (C3) and by updates A5.8.5 and A5.8.6 we have $\hat{x}_r^{\{\sigma+1\}} = \hat{x}_r^{\{\sigma\}}$ and $\bar{x}_r^{\{\sigma+1\}} = \bar{x}_r^{\{\sigma\}}$. Furthermore Observation 5.29 shows $J_r^{\{\underline{\pi}'\}} \subseteq \bar{J}^{[\pi']} \cup J^{[\pi']}$. Hence (iv) implies $\hat{y}^{\{\sigma+1\}}_{J_r^{\{\underline{\pi}'\}}} = \hat{y}^{\{\sigma\}}_{J_r^{\{\underline{\pi}'\}}}$. Consequently $\hat{x}_r^{\{\sigma+1\}} = \hat{x}_r^{\{\sigma\}}$ satisfies (C4) at $\sigma+1$ because no relevant information changed.

It remains to show (C1). By definition of J_r process η has no influence on the validity of (C1) unless there is some $j \in J^{[\eta]} \cap J_r$. But then $r \in R_{J^{[\eta]}} \setminus R^{[\eta]}$ by (ii) (because $r \in R^{[\pi']}$ by assumption). If the value of $\hat{y}_j^{\{\bar{\eta}+1\}}$ leads to a violation of (C1) for any such j, this j is included in $J_r^{\{\sigma+1\}}$ in the corresponding update A5.8.2. Therefore the subsequent test for (T1) in A5.8.3 fails because $j \in J_r^{\{\sigma+1\}} \cap J^{[\eta]}$ and η discards its results in A5.8.10 resulting in $\hat{y}^{\{\sigma+1\}} = \hat{y}^{\{\sigma\}}$. This $\hat{y}^{\{\sigma+1\}} = \hat{y}^{\{\sigma\}}$ satisfies (C1) by induction. □

Lemma 5.50 states a number of "invariant" properties that hold during the run of the algorithm. The next result collects a number of relations that are associated with the global indexes during the run of a certain process π, *i. e.* the indexes $\{\underline{\pi}, \dots, \bar{\pi}+1\}$.

Lemma 5.51 *(see Lemma 5.21)*
Given $\pi \in \Pi^{\{\infty\}}$, the following relations hold for all $\sigma \in \{\underline{\pi}, \dots, \bar{\pi}\}$.

$$\hat{y}^{\{\underline{\pi}\}}_{J^{[\pi]} \cup \bar{J}^{[\pi]}} = \hat{y}^{\{\sigma\}}_{J^{[\pi]} \cup \bar{J}^{[\pi]}}, \qquad \hat{w}^{\{\underline{\pi}\}}_{R^{[\pi]}} = \hat{w}^{\{\sigma\}}_{R^{[\pi]}}, \qquad \bar{w}^{\{\underline{\pi}\}}_{R^{[\pi]}} = \bar{w}^{\{\sigma\}}_{R^{[\pi]}}, \tag{5.84}$$

$$\hat{f}^{[\pi]}_{\bar{w}_r^{\{\sigma\}},r}(\hat{y}^{\{\sigma\}}_{J^{[\pi]}}) = \hat{f}_{\bar{w}_r^{\{\sigma\}},r}(\hat{y}^{\{\sigma\}}), \qquad \text{for all } r \in R^{[\pi]}, \tag{5.85}$$

$$f_r^{[\pi]}(\hat{y}^{\{\sigma\}}_{J^{[\pi]}}) = L_r(\hat{x}_r^{\{\underline{\pi}\}}, \hat{y}^{\{\underline{\pi}\}}) = L_r(\hat{x}_r^{\{\sigma\}}, \hat{y}^{\{\sigma\}}) = f_r(\hat{y}^{\{\sigma\}}), \quad \text{for all } r \in R^{[\pi]}. \tag{5.86}$$

For the step to $\bar{\pi}+1$,

$$\hat{y}^{\{\bar{\pi}+1\}}_{M \setminus J^{[\pi]}} = \hat{y}^{\{\bar{\pi}\}}_{M \setminus J^{[\pi]}}, \qquad \hat{w}^{\{\bar{\pi}+1\}}_{R \setminus R^{[\pi]}} = \hat{w}^{\{\bar{\pi}\}}_{R \setminus R^{[\pi]}}, \qquad \bar{w}^{\{\bar{\pi}+1\}}_{R \setminus R^{[\pi]}} = \bar{w}^{\{\bar{\pi}\}}_{R \setminus R^{[\pi]}}, \tag{5.87}$$

in particular,

$$\hat{y}^{\{\underline{\pi}\}}_{\bar{J}_r^{[\pi]}} = \hat{y}^{\{\sigma'\}}_{\bar{J}_r^{[\pi]}}, \qquad \text{for all } r \in R^{[\pi]},\ \sigma' \in \{\underline{\pi}, \dots, \bar{\pi}+1\}. \tag{5.88}$$

Furthermore, if π *does* not *fulfil the consistency check A5.8.3,*

$$\hat{y}^{\{\bar{\pi}+1\}}_{J^{[\pi]}} = \hat{y}^{\{\bar{\pi}\}}_{J^{[\pi]}}, \qquad \hat{w}^{\{\bar{\pi}+1\}}_{R^{[\pi]}} = \hat{w}^{\{\bar{\pi}\}}_{R^{[\pi]}}, \qquad \bar{w}^{\{\bar{\pi}+1\}}_{R^{[\pi]}} = \bar{w}^{\{\bar{\pi}\}}_{R^{[\pi]}}, \tag{5.89}$$

Otherwise, if π does *fulfil the consistency check with* $\hat{y}^{[\pi]} \in \mathbb{R}^{J^{[\pi]}}$ *and* $\bar{w}^{[\pi]} \in \operatorname{conv} W^{[\pi]}$ *being the final values of* π*, it holds*

$$\hat{y}^{\{\bar{\pi}+1\}}_{J^{[\pi]}} = \hat{y}^{[\pi]}, \qquad \hat{w}^{\{\bar{\pi}+1\}}_{R^{[\pi]}} = \hat{w}^{[\pi]}, \qquad \bar{w}^{\{\bar{\pi}+1\}}_{R^{[\pi]}} = \bar{w}^{[\pi]}, \tag{5.90}$$

$$f_r^{[\pi]}(\hat{y}^{[\pi]}) = L_r(\hat{x}_r^{\{\bar{\pi}+1\}}, \hat{y}^{\{\bar{\pi}+1\}}) = f_r(\hat{y}^{\{\bar{\pi}+1\}}), \qquad \text{for all } r \in R^{[\pi]}, \tag{5.91}$$

$$\hat{f}^{[\pi]}_{\bar{w}_r^{[\pi]},r}(\hat{y}^{[\pi]}) = \hat{f}_{\bar{w}_r^{\{\bar{\pi}+1\}},r}(\hat{y}^{\{\bar{\pi}+1\}}), \qquad \text{for all } r \in R^{[\pi]}, \tag{5.92}$$

$$g^{[\pi]}(\bar{x}^{\{\sigma\}}_{R^{[\pi]}}) = g(\bar{x}^{\{\sigma\}})_{J^{[\pi]}}, \qquad \text{for all } \sigma \in \{\underline{\pi}, \dots, \bar{\pi}+1\}. \tag{5.93}$$

Proof. We first prove (5.84)–(5.86) for $\sigma = \underline{\pi}$. The relations (5.84) are obvious for $\sigma = \underline{\pi}$. For (5.85) and (5.86) observe that $\pi \in \Pi^{\{\underline{\pi}+1\}}$, thus Lemma 5.50 (viii) for $\sigma = \underline{\pi}+1$ imply that $(\hat{y}^{\{\underline{\pi}+1\}}, \hat{x}_r^{\{\underline{\pi}+1\}}, \bar{x}_r^{\{\underline{\pi}+1\}}, J_r^{\{\underline{\pi}\}})$ is consistent for $r \in R^{[\pi]}$. Because π executes its subspace selection step at $\underline{\pi}$, the update of the global data in A5.9.3 implies $\hat{y}^{\{\underline{\pi}\}} = \hat{y}^{\{\underline{\pi}+1\}}$, $\hat{w}^{\{\underline{\pi}\}} = \hat{w}^{\{\underline{\pi}+1\}}$ and $\bar{w}^{\{\underline{\pi}\}} = \bar{w}^{\{\underline{\pi}+1\}}$, so the consistency also holds for $\sigma = \underline{\pi}$. Therefore we can apply Observation 5.42 for the data at $\sigma = \underline{\pi}$ proving (5.85) and (5.86).

Now consider $\sigma \in \{\underline{\pi}+1, \dots, \bar{\pi}\}$. Then we have $\pi \in \Pi^{\{\sigma\}}$ and we will apply Lemma 5.50 for these values of σ. (5.84) and the first two equations of (5.86) follow directly from (iv). (5.85) and (5.86) follow again from Observation 5.42, which can be applied by (viii).

Relations (5.87) follow from the fact that π executes its update step at $\bar{\pi}$, not changing new global values on $R \setminus R^{[\pi]}$ for $\bar{\pi}+1$ in A5.8.4 – A5.8.6.

(5.89) hold because the global data is not changed if the consistency test fails, see A5.8.10.

Now assume that the consistency test succeeds. Then (5.90) follow directly from the update in A5.8.4 – A5.8.6. By (5.88) and Lemma 5.50 (viii) we can apply Lemma 5.49 proving $(\hat{y}^{\{\bar{\pi}+1\}}, \hat{x}_r^{[\pi]}, \bar{x}_r^{[\pi]}, J_r^{\{\underline{\pi}\}})$ is weakly consistent for each $r \in R^{[\pi]}$ and Observation 5.42 shows (5.91) and (5.92).

In order to see the last relation (5.93) we prove that (5.81) holds for $\bar{x}^{\{\sigma\}}$, $\sigma \in \{\underline{\pi}, \dots, \bar{\pi}+1\}$, then the result follows from Observation 5.43. Consider an $r \in R_j \setminus R_j^{[\pi]}$ for some $j \in J^{[\pi]}$. The successful consistency test (T4) certifies $j \notin J_r^{\{\bar{\pi}+1\}}$ and $J_r^{\{\bar{\pi}+1\}} \supseteq J_r^{\{\sigma\}}$ by Lemma 5.50, (v). Relation (vii) implies that (C3) holds at σ, hence

$C_{j,r}\bar{x}_r^{\{\sigma\}} = 0$. This completes the proof. □

The previous result analyses the change of the global data during the run of a process π and when π itself interacts with the global data. The result concentrates on the subspace $J^{[\pi]}$ and the subproblems $R^{[\pi]}$ associated with π. But there is one further possibility on how π may influence the global data. This kind of influence is not present in the standard parallel bundle algorithm and is due to the update of the activity sets $J_r^{\{\sigma\}}$, $r \in R$, in the update step. In fact, π may change the activity sets of subproblems $R_{J^{[\pi]}} \setminus R^{[\pi]}$ in line A5.8.2, which are not directly related to its own subspace problem. The following result collects some of the global consistency relations stated in the previous lemmas and also takes care of this specific problematic.

Lemma 5.52 *(see Lemma 5.22)*
For $\sigma \in \Sigma$ it holds,

(i) $L_r(\hat{x}_r^{\{\sigma\}}, \hat{y}^{\{\sigma\}}) = f_r(\hat{y}^{\{\sigma\}})$ *for all* $r \in R$,

(ii) $\bar{w}^{\{\sigma\}} \in \operatorname{conv} W$,

(iii) $E^{\{\sigma\}} \subseteq E^{\{\sigma+1\}} \subseteq \{(i,j)\colon i,j \in M, i \neq j\}$.

Proof. We use induction on $\sigma \in \Sigma$. For $\sigma = 0$ the statement follows by the global initialisation step of the algorithm. So suppose now $\sigma + 1 \in \Sigma$ and the claim holds for $\sigma \in \Sigma$. By definition of Σ there is a process $\pi \in \Pi^{\{\sigma\}}$ that either executes its subspace selection step at $\sigma = \underline{\pi}$ or its update step at $\sigma = \bar{\pi}$. In the first case the update of the global data in A5.9.3 does not change any of the involved data, so the assertions remain true by induction.

Now assume $\sigma = \bar{\pi}$. We apply Lemma 5.51 and use the inductions hypothesis

- the global data at $\sigma + 1$ is consistent by Lemma 5.50 (vii), thus (i) follows from Observation 5.38,
- (5.87) and the fact $\bar{w}_r^{[\pi]} \in \operatorname{conv} W_r$, $r \in R^{[\pi]}$, as final value computed by Algorithm 5.7 prove (ii),
- the edge set is only increased in A5.8.8, so (iii) holds. □

In Section 5.4.4 we stated that we could not prove an analogous result to Lemma 5.13 that, given an appropriate choice of the parameter τ_1, there is always a one-dimensional subspace that provides sufficient decrease compared with the global predicted decrease. Equipped with the previous results we are now ready to state and prove this result.

Lemma 5.53 *(see Lemma 5.13)*
If $\tau_1 \in (0, \frac{1}{2}\min\{\frac{1}{|M|}, \frac{1}{|R|}\})$ *then there is always a* $j \in M$ *so that* $J = \{j\}$ *fulfils (5.67).*

Proof. The proof is similar to the proof of Lemma 5.13. Recall the definition (5.26) of $\Delta(\bar{w}, \hat{y})$

$$\Delta(\bar{w}, \hat{y}) = \sum_{r \in R} \left[f_r(\hat{y}) - \hat{f}_{\bar{w}_r, r}(\hat{y}) \right] + \frac{1}{u}\|g(\bar{x})\|^2.$$

At least one of both summands is greater than or equal to $\frac{1}{2}\Delta(\bar{w}, \hat{y})$. If this is true for the second summand, at least one of the terms $\frac{1}{u}g(\bar{x})_j^2$ is greater than or equal to $\frac{1}{2}\Delta(\bar{w}, \hat{y})\frac{1}{|M|}$. If this is true for the first one, at least one of the terms $f_r(\hat{y}) - \hat{f}_{\bar{w}_r, r}(\hat{y})$ is greater than or equal to $\frac{1}{2}\Delta(\bar{w}, \hat{y})\frac{1}{|R|}$. We show that in this case the set $J_r^{\{\sigma\}} \neq \emptyset$. Then we can choose $J = \{j\}$ for any $j \in J_r^{\{\sigma\}}$ and then $r \in R^{[\pi]}$. Note that in contrast to the standard parallel bundle method it is now possible that $J_{r'}^{\{\sigma\}} = \emptyset$ for some $r' \in R$ and therefore no constraint can be selected that enforces r' to be contained in $R_J^{[\pi]}$ (the standard parallel bundle method always used $R_J = R_J^{[\pi]}$).

Suppose, for a contradiction, $J_r^{\{\sigma\}} = \emptyset$. Then we know by (v) that $J_r^{\{0\}} = \emptyset$ and $f_r(\hat{y}^{\{0\}}) = \hat{f}_{\bar{w}^{\{0\}}, r}(\hat{y}^{\{0\}})$ by the initialisation step of the algorithm. Furthermore $r \notin R^{[\eta]}$ for any $\eta \in \underline{\Pi}^{\{\sigma\}}$ because otherwise $J_r^{\{\sigma\}} \supseteq J_r^{\{\underline{\eta}\}} \neq \emptyset$, *i. e.* r has never been selected by a process before. This implies $\bar{w}^{\{\sigma\}} = \bar{w}^{\{0\}}$ because the global aggregate can only be modified if the subproblem r has been selected by some process. Because (C3) holds we know $C_{J_r, r}\bar{x}^{\{\sigma\}} = 0$, hence by definition (5.20) of $\hat{f}_{\bar{w}, r}(y)$ it is

$$\hat{f}_{\bar{w}^{\{0\}}, r}(\hat{y}^{\{0\}}) = \bar{l}_r^{\{0\}} = \bar{l}_r^{\{\sigma\}} = \hat{f}_{\bar{w}^{\{\sigma\}}, r}(\hat{y}^{\{\sigma\}}),$$

i. e. the model value in the centre remains unchanged. Furthermore by Observation 5.38 and (C4) the value in the centre is $f_r(\hat{y}^{\{\sigma\}}) = h_r(\hat{x})$ for $\hat{x} \in \operatorname{Argmax}\{h_r(x_r) \colon x_r \in \mathcal{X}_r\}$. Consequently, as long as $J_r^{\{\sigma\}} = \emptyset$ the function value in the centre does not change although the centre may have changed, *i. e.* $f_r(\hat{y}^{\{0\}}) = h_r(\hat{x}_r) = f_r(\hat{y}^{\{\sigma\}})$. Putting all together neither the function value nor the model value in centre changed, consequently

$$f_r(\hat{y}^{\{\sigma\}}) - \hat{f}_{\bar{w}^{\{\sigma\}}, r}(\hat{y}^{\{\sigma\}}) = f_r(\hat{y}^{\{0\}}) - \hat{f}_{\bar{w}^{\{0\}}, r}(\hat{y}^{\{0\}}) = 0,$$

a contradiction. This completes the proof. □

The last result of this section is a direct analogue to Observation 5.23. The proof is equivalent, one only has to convince oneself that processes, whose consistency test fail, do not have any influence on the correctness of the arguments of the proof.

Observation 5.54 *(see Observation 5.23)*
For each $\sigma \in \Sigma$ the following statements are equivalent:

(i) $\sigma = \max \Sigma$,

(ii) $\Pi^{\{\sigma\}} = \Pi^{\{\sigma+1\}}$,

(iii) the algorithm terminated *with σ being the last global index,*

(iv) $\Delta(\bar{w}^{\{\sigma\}}, \hat{y}^{\{\sigma\}}) \leq \varepsilon(|f(\hat{y}^{\{\sigma\}})| + 1)$ *and* $\Delta(\bar{w}^{\{\sigma'\}}, \hat{y}^{\{\sigma'\}}) > \varepsilon(|f(\hat{y}^{\{\sigma'\}})| + 1)$ *for all* $\sigma' < \sigma$.

In particular, if $\max \Sigma = \infty$ *then the algorithm does not terminate and* $\Delta(\bar{w}^{\langle\sigma\rangle}, \hat{y}^{\langle\sigma\rangle}) > \varepsilon(|f(\hat{y}^{\langle\sigma\rangle})| + 1)$ *for all* $\sigma \in \mathbb{N}_0 = \Sigma$ *and vice versa.*

Proof. Analogous to proof of Observation 5.23. □

5.4.9 Convergence Analysis

For the convergence analysis the same steps work out as for the first variant. It will be convenient to collect all processes, that satisfied the consistency test in A5.8.3, in the set

$$\Gamma := \{\pi \in \bar{\Pi}^{\{\infty\}} : \pi \text{ satisfies the consistency test}\}. \tag{5.94}$$

The reason is that processes that do *not* satisfy this test do not modify the global primal and dual values $\bar{w}^{\{\sigma\}}$ and $\hat{y}^{\{\sigma\}}$ by (5.89). We start by proving that the global progress matches that of a single process.

Lemma 5.55 *(see Lemma 5.24)*
For $\pi \in \Gamma$

$$0 \leq f^{[\pi]}(\hat{y}^{\{\bar{\pi}\}}_{J^{[\pi]}}) - f^{[\pi]}(\hat{y}^{\{\bar{\pi}+1\}}_{J^{[\pi]}}) = f(\hat{y}^{\{\bar{\pi}\}}) - f(\hat{y}^{\{\bar{\pi}+1\}}),$$

i. e., the global progress achieved when π stores its subspace solution in the global data is exactly the progress made by π on its subspace problem. In particular, the sequence $(f(\hat{y}^{\{\sigma\}}))_{\sigma\in\Sigma}$ is non-increasing.

Proof. We apply the results of Lemma 5.51. First we prove the right-hand equation. It holds

$$\begin{aligned} f^{[\pi]}(\hat{y}^{\{\underline{\pi}\}}_{J^{[\pi]}}) - f^{[\pi]}(\hat{y}^{\{\bar{\pi}+1\}}_{J^{[\pi]}}) &\overset{(5.74)}{=} b^T_{J^{[\pi]}}(\hat{y}^{\{\underline{\pi}\}}_{J^{[\pi]}} - \hat{y}^{\{\bar{\pi}+1\}}_{J^{[\pi]}}) + \sum_{r\in R^{[\pi]}} \Big[f^{[\pi]}_r(\hat{y}^{\{\underline{\pi}\}}_{J^{[\pi]}}) - f^{[\pi]}_r(\hat{y}^{\{\bar{\pi}+1\}}_{J^{[\pi]}})\Big] \\ &\overset{(5.84)}{=} b^T_{J^{[\pi]}}(\hat{y}^{\{\bar{\pi}\}}_{J^{[\pi]}} - \hat{y}^{\{\bar{\pi}+1\}}_{J^{[\pi]}}) + \sum_{r\in R^{[\pi]}} \Big[f^{[\pi]}_r(\hat{y}^{\{\bar{\pi}\}}_{J^{[\pi]}}) - f^{[\pi]}_r(\hat{y}^{\{\bar{\pi}+1\}}_{J^{[\pi]}})\Big] \\ &\overset{\text{Lemma } 5.51}{=} b^T(\hat{y}^{\{\bar{\pi}\}} - \hat{y}^{\{\bar{\pi}+1\}}) + \sum_{r\in R} \Big[f_r(\hat{y}^{\{\bar{\pi}\}}) - f_r(\hat{y}^{\{\bar{\pi}+1\}})\Big] \\ &\overset{(5.17)}{=} f(\hat{y}^{\{\bar{\pi}\}}) - f(\hat{y}^{\{\bar{\pi}+1\}}). \end{aligned}$$

If π does not pass the consistency tests then it does not change the global centre, *i. e.* $\hat{y}^{\{\bar{\pi}\}} = \hat{y}^{\{\bar{\pi}+1\}}$ and the inequality holds trivially, so assume π passes the consistency tests. The initial centre of the subspace problem is $\hat{y}^{\{\underline{\pi}\}}_{J^{[\pi]}} \overset{(5.84)}{=} \hat{y}^{\{\bar{\pi}\}}_{J^{[\pi]}}$, the final centre is $\hat{y}^{\{\bar{\pi}+1\}}_{J^{[\pi]}}$. If π stops because of condition [X1], *i. e.*, no descent step occurs, then its centre remains unchanged, *i. e.*, $\hat{y}^{\{\bar{\pi}+1\}}_{J(\pi)} = \hat{y}^{[\pi]} = \hat{y}^{\{\underline{\pi}\}}_{J(\pi)}$ and the inequality holds. If otherwise π stops because of condition [X2] and passes the consistency tests, then π performs a descent step, implying the inequality as well.

It remains to show that the sequence $(f(\hat{y}^{\{\sigma\}}))_\sigma$ is non-increasing. For any $\sigma \in \Sigma$ there is a process $\pi \in \underline{\Pi}^{\{\infty\}}$ with $\sigma \in \{\underline{\pi}, \bar{\pi}\}$. If $\sigma = \bar{\pi}$ and $\pi \in \Gamma$ then the result above shows $f(\hat{y}^{\{\sigma\}}) \geq f(\hat{y}^{\{\sigma+1\}})$. Otherwise π executes either its subspace selection step at $\sigma = \underline{\pi}$ or $\sigma = \underline{\pi} \wedge \pi \notin \Gamma$, which means it does not satisfy the consistency check. In the case $\sigma = \underline{\pi}$ process π does not change the global centre in A5.9.3 and in the case $\sigma = \bar{\pi}$ and $\pi \notin \Gamma$ Lemma 5.51 shows $\hat{y}^{\{\sigma\}} = \hat{y}^{\{\sigma+1\}}$, hence $f(\hat{y}^{\{\sigma\}}) = f(\hat{y}^{\{\sigma+1\}})$. □

For the next step we first have to establish the required relations between the predicted decrease of the global data and that of a process. We use in the following the short notation for each $\sigma \in \mathbb{N}_0$ and each subspace $J \subseteq M$, see (5.62), (5.64) and (5.65)

$\Delta^{\{\sigma\}}$ $$\Delta^{\{\sigma\}} := \Delta(\bar{w}^{\{\sigma\}}, \hat{y}^{\{\sigma\}}),$$

$\Delta^{\{\sigma\}}_\pi$ $$\Delta^{\{\sigma\}}_\pi := \Delta_\pi(\bar{w}^{\{\sigma\}}, \hat{y}^{\{\sigma\}}),$$

$\bar{\Delta}^{\{\sigma\}}_\pi$ $$\bar{\Delta}^{\{\sigma\}}_\pi := \bar{\Delta}_\pi(\bar{w}^{\{\sigma\}}, \hat{y}^{\{\sigma\}}),$$

$\delta^{\{\sigma\}}_\pi$ $$\delta^{\{\sigma\}}_\pi := \delta_\pi(\bar{w}^{\{\sigma\}}).$$

Lemma 5.56 *For $\pi \in \Gamma$*

$$\Delta^{\{\bar{\pi}\}}_\pi = \Delta^{\{\underline{\pi}\}}_\pi = \Delta^{[\pi]}(\bar{w}^{\{\underline{\pi}\}}_{R^{[\pi]}}, \hat{y}^{\{\underline{\pi}\}}_{J^{[\pi]}}), \tag{5.95}$$

$$\bar{\Delta}^{\{\bar{\pi}+1\}}_\pi = \bar{\Delta}^{\{\bar{\pi}\}}_\pi, \tag{5.96}$$

$$\Delta^{\{\bar{\pi}+1\}}_\pi = \Delta^{[\pi]}(\bar{w}^{\{\bar{\pi}+1\}}_{R^{[\pi]}}, \hat{y}^{\{\bar{\pi}+1\}}_{J^{[\pi]}}). \tag{5.97}$$

Proof. We will use the results of Lemma 5.51 in the following.

$$\begin{aligned}
\Delta_{\pi}^{\{\bar{\pi}\}} &\overset{(5.64)}{=} \sum_{r\in R^{[\pi]}} \left[f_r(\hat{y}^{\{\bar{\pi}\}}) - \hat{f}_{\bar{w}_r^{\{\bar{\pi}\}},r}(\hat{y}^{\{\bar{\pi}\}}) \right] + \frac{1}{u}\|g(\bar{x}^{\{\bar{\pi}\}})_{J^{[\pi]}}\|^2 \\
&\overset{(5.85),(5.86),(5.93)}{=} \sum_{r\in R^{[\pi]}} \left[f_r^{[\pi]}(\hat{y}_{J^{[\pi]}}^{\{\bar{\pi}\}}) - \hat{f}^{[\pi]}_{\bar{w}_r^{\{\bar{\pi}\}},r}(\hat{y}_{J^{[\pi]}}^{\{\bar{\pi}\}}) \right] + \frac{1}{u}\|g^{[\pi]}(\bar{x}_{R^{[\pi]}}^{\{\bar{\pi}\}})\|^2 \\
&\overset{(5.84)}{=} \sum_{r\in R^{[\pi]}} \left[f_r^{[\pi]}(\hat{y}_{J^{[\pi]}}^{\{\underline{\pi}\}}) - \hat{f}^{[\pi]}_{\bar{w}_r^{\{\underline{\pi}\}},r}(\hat{y}_{J^{[\pi]}}^{\{\underline{\pi}\}}) \right] + \frac{1}{u}\|g^{[\pi]}(\bar{x}_{R^{[\pi]}}^{\{\underline{\pi}\}})\|^2 \\
&\overset{(5.85),(5.86),(5.93)}{=} \sum_{r\in R^{[\pi]}} \left[f_r(\hat{y}^{\{\underline{\pi}\}}) - \hat{f}_{\bar{w}_r^{\{\underline{\pi}\}},r}(\hat{y}^{\{\underline{\pi}\}}) \right] + \frac{1}{u}\|g(\bar{x}^{\{\underline{\pi}\}})_{J^{[\pi]}}\|^2 \\
&\overset{(5.64)}{=} \Delta_{\pi}^{\{\underline{\pi}\}}.
\end{aligned}$$

To complete (5.95) note that by (5.83) the third line is in fact $\Delta^{[\pi]}(\bar{w}_{R^{[\pi]}}^{\{\underline{\pi}\}}, \hat{y}_{J^{[\pi]}}^{\{\underline{\pi}\}})$.

In order to prove (5.96) we first show (with $\hat{J}^{[\pi]}$ as defined in (5.62))

$$C_{j,r}\bar{x}_r^{\{\bar{\pi}\}} = C_{j,r}\bar{x}_r^{\{\bar{\pi}+1\}} \quad \text{for } j \in M \setminus (J^{[\pi]} \cup \hat{J}^{[\pi]}), r \in R. \tag{5.98}$$

For $r \in R \setminus R^{[\pi]}$ this is an immediate consequence of (5.87). For $r \in R^{[\pi]}$ consider some fixed $j \in M \setminus (J^{[\pi]} \cup \hat{J}^{[\pi]})$ and recall that, by (5.62) and Lemma 5.50 (v), $J_r^{\{\bar{\pi}\}} \subseteq J_r^{\{\bar{\pi}+1\}} \subseteq J^{[\pi]} \cup \hat{J}^{[\pi]}$. Using Lemma 5.50 (vii) we get $C_{j,r}\bar{x}_r^{\{\bar{\pi}\}} \overset{(C3)}{=} 0 \overset{(C3)}{=} C_{j,r}\bar{x}_r^{\{\bar{\pi}+1\}}$. This proves (5.98).

With (5.98) and by recalling the definition (5.18) of $g(x)$ we obtain as a first ingredient for (5.96)

$$\frac{1}{u}\|g(\bar{x}^{\{\bar{\pi}\}})_{M\setminus(J^{[\pi]}\cup\hat{J}^{[\pi]})}\|^2 = \frac{1}{u}\|g(\bar{x}^{\{\bar{\pi}+1\}})_{M\setminus J^{[\pi]}\cup\hat{J}^{[\pi]}}\|^2.$$

Next we want to show

$$\hat{f}_{\bar{w}_r^{\{\bar{\pi}\}},r}(\hat{y}^{\{\bar{\pi}\}}) = \hat{f}_{\bar{w}_r^{\{\bar{\pi}+1\}},r}(\hat{y}^{\{\bar{\pi}+1\}}) \quad \text{for } r \in R \setminus R^{[\pi]}.$$

By definition (5.20) this requires to show

$$\bar{l}_r^{\{\bar{\pi}\}} = \bar{l}_r^{\{\bar{\pi}+1\}}$$

and

$$(\hat{y}_{J_r}^{\{\bar{\pi}\}})^T C_{J_r,r}\bar{x}_r^{\{\bar{\pi}\}} = (\hat{y}_{J_r}^{\{\bar{\pi}+1\}})^T C_{J_r,r}\bar{x}_r^{\{\bar{\pi}+1\}}, \qquad \text{for } r \in R \setminus R^{[\pi]}.$$

Because of (5.87) the only possible difference might be caused by $\hat{y}_j^{\{\bar{\pi}\}} \neq \hat{y}_j^{\{\bar{\pi}+1\}}$ for some $j \in J^{[\pi]}$. For a fixed $j \in J^{[\pi]}$ and $r \in R_j \setminus R^{[\pi]}$, however, the successful consistency test (T4) at $\underline{\pi}$ asserts $j \notin J_r^{\{\bar{\pi}+1\}} \supseteq J_r^{\{\bar{\pi}\}}$. Therefore (C3) ensures $C_{j,r}\bar{x}_r^{\{\bar{\pi}\}} = 0 = C_{j,r}\bar{x}_r^{\{\bar{\pi}+1\}}$.

For the final ingredient fix $r \in R \setminus R^{[\pi]}$ and observe that $\bar{x}_r^{\{\bar{\pi}\}} = \bar{x}_r^{\{\bar{\pi}+1\}}$ by Lemma 5.51 (5.87). Because $\pi \in \Gamma$ the successful consistency test (T4) implies $J_r^{\{\bar{\pi}+1\}} \cap J^{[\pi]} = \emptyset$, hence $\hat{y}_{J_r^{\{\bar{\pi}+1\}}}^{\{\bar{\pi}\}} = \hat{y}_{J_r^{\{\bar{\pi}+1\}}}^{\{\bar{\pi}+1\}}$ by (5.87). Using Lemma 5.50 (vii) we know that $(\hat{y}^{\{\bar{\pi}\}}, \hat{x}_r^{\{\bar{\pi}\}}, \bar{x}_r^{\{\bar{\pi}\}}, J_r^{\{\bar{\pi}\}})$

and $(\hat{y}^{\{\bar{\pi}+1\}}, \hat{x}_r^{\{\bar{\pi}+1\}}, \bar{x}_r^{\{\bar{\pi}+1\}}, J_r^{\{\bar{\pi}+1\}})$ are consistent. Furthermore Lemma 5.50 (v) and Observation 5.38 imply that $(\hat{y}^{\{\bar{\pi}\}}, \hat{x}_r^{\{\bar{\pi}\}}, \bar{x}_r^{\{\bar{\pi}\}}, J_r^{\{\bar{\pi}+1\}})$ is consistent, too, and

$$\begin{aligned} f_r(\hat{y}^{\{\bar{\pi}\}}) &\overset{(5.66)}{=} h_r(\hat{x}_r^{\{\bar{\pi}\}}) - (\hat{y}^{\{\bar{\pi}\}}_{J_r^{\{\bar{\pi}+1\}}})^T C_{J_r^{\{\bar{\pi}+1\}},r} \hat{x}_r^{\{\bar{\pi}\}} \\ &= h_r(\hat{x}_r^{\{\bar{\pi}+1\}}) - (\hat{y}^{\{\bar{\pi}+1\}}_{J_r^{\{\bar{\pi}+1\}}})^T C_{J_r^{\{\bar{\pi}+1\}},r} \hat{x}_r^{\{\bar{\pi}+1\}} \\ &\overset{(5.66)}{=} f_r(\hat{y}^{\{\bar{\pi}+1\}}). \end{aligned}$$

The last equation (5.97) is a direct consequence of the definitions (5.64) and (5.83) together with Lemma 5.51. □

From now on, the analysis follows exactly the same path as that of the standard parallel bundle method. The proofs match the first one verbatim up to small adaptions due to the presence of processes not passing the consistency test.

Lemma 5.57 *(see Lemma 5.25)*
Suppose an infinite number of descent steps occurs and f is bounded from below. Then

$$\liminf_{\sigma \in \mathbb{N}_0} \Delta^{\{\sigma\}} \to 0.$$

Proof. Let $\pi \in \Gamma$ be a process that satisfies the consistency check A5.8.3 and for which a descent step occurs, *i. e.*, π is stopped because of condition [X2]. The subspace selection condition (5.67) implies $\Delta^{\{\underline{\pi}\}} \leq \frac{1}{\tau_1} \Delta_\pi^{\{\underline{\pi}\}}$. Together with Observation 5.46 and Lemma 5.55 we get

$$\Delta^{\{\underline{\pi}\}} \leq \frac{1}{\tau_1} \Delta_\pi^{\{\underline{\pi}\}} \leq \frac{1}{\tau_1 \tau_2 \varrho} \left(f^{[\pi]}(\hat{y}^{\{\underline{\pi}\}}_{J^{[\pi]}}) - f^{[\pi]}(\hat{y}^{\{\bar{\pi}+1\}}_{J^{[\pi]}}) \right) = \frac{1}{\tau_1 \tau_2 \varrho} \left(f(\hat{y}^{\{\bar{\pi}\}}) - f(\hat{y}^{\{\bar{\pi}+1\}}) \right).$$

Because f is bounded from below and the sequence $(f(\hat{y}^{\{\sigma\}}))_\sigma$ is non-increasing by Lemma 5.55, the right-hand side of the inequality above converges to zero. □

Lemma 5.58 *(see Lemma 5.26)*
Assume there is only a finite number of descent steps and $\varepsilon = 0$, then

$$\lim_{\sigma \in \Sigma} \Delta^{\{\sigma\}} = 0.$$

Proof. If $|\Sigma| < \infty$, then the statement holds by Observation 5.54. Therefore we may assume $|\Sigma| = \infty$. In this case $\Pi^{\{\sigma\}} \subseteq \bar{\Pi}^{\{\infty\}}$ for all $\sigma \in \Sigma$ by Observation 5.45.

Observation 5.54 also implies $\Delta^{\{\sigma\}} > 0$ for all $\sigma \in \Sigma$ and by Lemma 5.52 the edge set $E^{\{\sigma\}}$ of the dependency graph can only be increased. Similarly the activity sets $J_r^{\{\sigma\}} \subseteq M$ are only increased by Lemma 5.50 (v). Because M is a finite set there must be a $\underline{\sigma} \in \Sigma$ such that for each $\sigma \geq \underline{\sigma}$ we have $E^{\{\sigma\}} = E^{\{\underline{\sigma}\}}$, $J_r^{\{\sigma\}} = J_r^{\{\underline{\sigma}\}}$, $r \in R$, and all processes π with $\bar{\pi} > \underline{\sigma}$ do not perform a descent step. Put $\bar{\sigma} := \max\big(\{\underline{\sigma}\} \cup \{\bar{\pi}' \colon \pi' \in \Pi^{\{\underline{\sigma}\}}\}\big)$. Then for π with $\bar{\pi} > \bar{\sigma}$ we have $\underline{\pi} > \underline{\sigma}$.

Let $\sigma > \underline{\sigma}$, then there is a process π such that $\sigma \in \{\underline{\pi}, \bar{\pi}\}$. If $\sigma = \underline{\pi}$ then π executes a subspace selection step and does not change $\Delta^{\{\sigma\}}$ in A5.9.3, hence $\Delta^{\{\underline{\pi}\}} = \Delta^{\{\underline{\pi}+1\}}$. So assume $\sigma = \bar{\pi}$. If $\pi \notin \Gamma$ then the global data does not change and again $\Delta^{\{\bar{\pi}\}} = \Delta^{\{\bar{\pi}+1\}}$. If $\pi \in \Gamma$ then because $\sigma > \underline{\sigma}$ process π satisfied condition [X1] and $E^{\{\sigma+1\}} = E^{\{\sigma\}}$. Therefore Algorithm 5.8 did not increase the edge set and by the test in A5.8.7 it holds

$$\delta_\pi^{\{\bar{\pi}+1\}} - \delta_\pi^{\{\bar{\pi}\}} \leq \tau_3 \Delta_\pi^{\{\underline{\pi}\}}.$$

Invoking Observation 5.32 twice for the subspace $J^{[\pi]}$ of π but once for the data of $\bar{\pi}$ and once for $\bar{\pi}+1$ yields the relations

$$\begin{aligned}\Delta^{\{\bar{\pi}\}} &= \Delta_\pi^{\{\bar{\pi}\}} + \delta_\pi^{\{\bar{\pi}\}} + \bar{\Delta}_\pi^{\{\bar{\pi}\}},\\ \Delta^{\{\bar{\pi}+1\}} &= \Delta_\pi^{\{\bar{\pi}+1\}} + \delta_\pi^{\{\bar{\pi}+1\}} + \bar{\Delta}_\pi^{\{\bar{\pi}+1\}}.\end{aligned}$$

We claim that $\Delta^{\{\bar{\pi}+1\}} \leq (1-\tau)\Delta^{\{\bar{\pi}\}}$ for some constant $0 < \tau < 1$ independent of π. Indeed, by Lemma 5.56 (5.95) we have $\Delta_\pi^{\{\underline{\pi}\}} = \Delta_\pi^{\{\bar{\pi}\}}$ and (5.96) gives $\bar{\Delta}_\pi^{\{\bar{\pi}\}} = \bar{\Delta}_\pi^{\{\bar{\pi}+1\}}$. The subspace selection condition (5.67) asserts $\Delta_\pi^{\{\underline{\pi}\}} \geq \tau_1 \Delta^{\{\underline{\pi}\}}$ and stopping condition [X1] implies $\Delta_\pi^{\{\bar{\pi}+1\}} \overset{(5.97)}{=} \Delta^{[\pi]}(\hat{y}_{J^{[\pi]}}^{\{\bar{\pi}+1\}}, \bar{w}_{R^{[\pi]}}^{\{\bar{\pi}+1\}}) < \tau_2 \Delta_\pi^{\{\underline{\pi}\}}$. This yields

$$\begin{aligned}&\Delta^{\{\bar{\pi}\}} - \Delta^{\{\bar{\pi}+1\}}\\ &\quad= (\Delta_\pi^{\{\bar{\pi}\}} - \Delta_\pi^{\{\bar{\pi}+1\}}) + (\bar{\Delta}_\pi^{\{\bar{\pi}\}} - \bar{\Delta}_\pi^{\{\bar{\pi}+1\}}) + (\delta_\pi^{\{\bar{\pi}\}} - \delta_\pi^{\{\bar{\pi}+1\}})\\ &\quad= (\Delta_\pi^{\{\underline{\pi}\}} - \Delta_\pi^{\{\bar{\pi}+1\}}) + (\delta_\pi^{\{\bar{\pi}\}} - \delta_\pi^{\{\bar{\pi}+1\}})\\ &\quad\geq (1 - \tau_2 - \tau_3)\Delta_\pi^{\{\underline{\pi}\}}\\ &\quad\geq \underbrace{\tau_1(1-\tau_2-\tau_3)}_{=:\tau \in (0,1)} \Delta^{\{\underline{\pi}\}}.\end{aligned} \tag{5.99}$$

Note that this shows $\Delta^{\{\bar{\pi}\}} - \Delta^{\{\bar{\pi}+1\}} \geq 0$ for all $\bar{\pi} = \sigma \geq \underline{\sigma}$. Together with $\Delta^{\{\underline{\pi}\}} = \Delta^{\{\underline{\pi}+1\}}$ (see above) we get therefore that the sequence $(\Delta^{\{\sigma\}})_{\sigma \geq \underline{\sigma}}$ is non-increasing.

Now assume $\bar{\pi} = \sigma > \bar{\sigma}$, then we have $\underline{\pi} \geq \underline{\sigma}$ and thus $\Delta^{\{\underline{\pi}\}} \geq \Delta^{\{\bar{\pi}\}}$. From (5.99) we obtain

$$\Delta^{\{\bar{\pi}\}} - \Delta^{\{\bar{\pi}+1\}} \geq \tau \Delta^{\{\underline{\pi}\}} \geq \tau \Delta^{\{\bar{\pi}\}}$$

and so

$$\Delta^{\{\bar{\pi}+1\}} \leq (1-\tau)\Delta^{\{\bar{\pi}\}}.$$

Together with the case $\sigma = \underline{\pi}$ above we get $\lim_{\sigma \in \mathbb{N}_0} \Delta^{\{\sigma\}} = 0$, which completes the proof. □

Corollary 5.59 *(see Corollary 5.27)*
If f is bounded from below and $\varepsilon = 0$, the predicted decrease $\Delta^{\{\sigma\}} = f(\hat{y}^{\{\sigma\}}) - \hat{f}_{\bar{w}^{\{\sigma\}}}(\hat{y}^{\{\sigma\}}) + \frac{1}{u}\|g(\bar{x}^{\{\sigma\}})\|^2$ goes to zero for an appropriate subsequence $\Sigma^* \subseteq \Sigma$. In particular, $f(\hat{y}^{\{\sigma\}}) - \hat{f}_{\bar{w}^{\{\sigma\}}}(\hat{y}^{\{\sigma\}})$ and $\|g(\bar{x}^{\{\sigma\}})\|$ go to zero, too, for the subsequence Σ^*.

Proof. Analogous to the proof of Corollary 5.27. □

Theorem 5.60 *Suppose $\emptyset \neq \operatorname{Argmin} f$ is bounded. Then for an appropriate subsequence $\Sigma^* \subseteq \Sigma$ the sequences $(\hat{y}^{\{\sigma\}})_{\sigma\in\Sigma^*}$ and $(\bar{x}^{\{\sigma\}})_{\sigma\in\Sigma^*}$ that are generated by the parallel bundle algorithm have the following properties.*

(i) each accumulation point of $(\hat{y}^{\{\sigma\}})_{\sigma\in\Sigma^*}$ is an optimal solution of (LD),

*(ii) each accumulation point of $(\bar{x}^{\{\sigma\}})_{\sigma\in\Sigma^*}$ is an optimal solution of (*conv *P).*

Proof. Analogous to the proof of Theorem 5.28 □

We finish the section with a remark about primal cutting planes, see Section 5.2.1.

Remark 5.61 Although we did not mention it explicitly, the extended version of the algorithm can be used with primal cutting planes as well. Indeed, assume that a constraint $j \in M$ is not contained in any activity set, $\forall r \in R\colon j \notin J_r^{\{\sigma\}}$. Then this constraint will not be contained in any selected subspace. Only when j is added to some activity set it has to be considered at all. In fact, only if j is selected for the first time in some subspace it has to be considered at all, which happens only if the corresponding coordinate of the aggregated subgradient is non-zero (*i. e.*, if the aggregated primal solution violates this constraint, otherwise this constraint will never contribute to the expected progress $\Delta^{\langle\sigma\rangle}$). However, the consistency of the global data requires additional care when dealing with primal cutting planes. Furthermore, in the parallel setting the constraint may be added in one process while another process is working on other subproblems interacting with this constraint, so the algorithm has to handle these situations as well.

5.5 Parallel Bundle Method and the TTP

In this section we shortly discuss how our practical application, the TTP (see Chapter 2), works together with the parallel bundle method. According to the descriptions in Chapter 3, particularly Section 3.3, Lagrangian relaxation can be employed that leads to a problem

of type (LD), with one exception: the coupling constraints in both of our TTP models contain *inequality* constraints. Fortunately, this is not problem at all (see Remark 5.1), we simply have to solve the subspace problems with a bundle method that supports box constraints (see Remark 5.5).

In practice, we always want to employ the extended version of the parallel bundle algorithm of Section 5.4. The reason is that some constraints typically couple a lot of trains. This is particularly true for capacity constraints (2.6): each train that can potentially use a certain track $a \in A^I$ at a time step $t \in T$ has a non-zero coefficient in the corresponding constraint, although the train would never run at that time over a (*e. g.* the train runs early in the morning but t is a time step in the afternoon). Similar relations may hold for clique inequalities (2.7) if the number of trains in a clique are sufficiently large (sufficiently large means ≥ 3, then a conflict between two of the trains would couple them with the third train even if the latter is not actually involved in that conflict). In contrast, configuration constraints (2.8) only couple a certain train with a configuration network, so they have a structure that would also be sufficient for the simple parallel bundle method.

In order to be able to use the extended version, we have to ensure that we can easily verify (C1)–(C4). (C2)–(C4) are handled by the algorithm in all cases, but (C1) is hard to check in general. Fortunately, the structure of the coupling constraints makes this condition easy to check in our case. All coupling constraints have only non-negative coefficients and the primal variables are always incidence vectors, *i. e.* they are non-negative, too (see models (TTP-clq) and (TTP-cfg)). Thus we can apply the strategy of Remark 5.34.

Finally we want to mention that the TTP problems of the size we consider typically contain a large number of potential constraints. Thus, we would have to combine the parallel bundle method with primal cutting planes (see Section 5.2.1). Using primal cutting planes seems not to be easy with the simple parallel bundle method, but appears to be implementable with the extended version. However, our current implementation does not support primal cutting planes, yet, so we restricted our numerical tests to artificial test instances (see Section 6.3).

6 Numerical Tests

In this chapter we present the results of some numerical tests for the techniques developed in thesis. Although we also provide a few experiments for solving the real world instances of the TTP that motivated our developments, these should not be considered an in depth numerical study for the problems. Instead they should be thought of an illustration how our techniques can be applied in practice.

This chapter is divided into four sections. In Section 6.1 we present the real world instances from *Deutsche Bahn*. These instances will be used afterwards in Section 6.2 to compare the time expanded networks with and without dynamic graph generation. In Section 6.3 we present some experimental results for the parallel bundle algorithm. We have not applied the parallel bundle algorithm to our real-world instances, yet. The reason is that this would require further developments, *e. g.*, support for primal column generation and primal cutting planes. In particular, primal column generation would be necessary to combine the parallel bundle method with dynamic graph generation (generating new arcs in the dynamic networks corresponds to primal column generation in the Lagrangian relaxation). Therefore we only generated (static) random test instances of a simpler TTP problem. Finally in Section 6.4 we give some numerical results for our real world problem.

6.1 The Real World Problem

The largest data set that we consider is part of the German railway network of *Deutsche Bahn*, the so called "South-West-subnetwork". It comprises roughly Baden-Wuerttemberg, which is about 10% of the whole network. The data does not represent the microscopic railway network. Instead, the network is a macroscopic representation of the real network, where several nodes and tracks have been contracted to single nodes and tracks. For example, the microscopic network models each single track in a station, whereas our network represents many stations as a single node. Only very large stations may be represented by several nodes if otherwise their structure would be oversimplified. Similarly, in practice each track is divided into single blocking sections and at most one train is allowed to be in a certain blocking section at the same time. In our network several successive blocking sections have been collected in a single arc, the blocking restrictions are represented by headway times. Furthermore the data for the trains, in particular their routes, has been extracted from existing timetables. These trains and all related data (*e. g.*, running times) have been "projected" on the macroscopic network. The trains have been partitioned into 16 train types that have similar driving characteristics. In reality, there are much more different train types in use (depending not only on the locomotive but also on other parameters like length and weight).

instance	$\lvert V^I \rvert$	$\lvert A^I \rvert$
RM	445	744
SW	5512	12644

Table 6.1: Sizes of the infrastructure network. "RM" denotes the instances on the *Rhein-Main-Schiene*, "SW" the instances on the whole south-west subnetwork.

All these construction/contraction steps of the network have been done beforehand by *Deutsche Bahn*. The data that we can access consists of the contracted network together with the specification of the trains, running times, headway times and capacities according to the problem description of Chapter 2. This network of our data is shown in Figure 6.1. Besides the full instance we also extracted a smaller network that consists of the major long-distance and freight route along the river Rhine, the so called "Rhein-Main-Schiene". Indeed, this instance consists of all trains that run on the "Rhein-Main-Schiene" and all arcs and nodes that are used by these trains. The data comprises the trains of one day running in the network. We extracted 14 instances, each covering a time window of six hours (this was requested by *Deutsche Bahn*) starting at full hours from 3:00 until 16:00. We denote the instances of "Rhein-Main-Schiene" by *RM-h* and those of the South-West subnetwork by *SW-h* where $h \in \{3, 4, \ldots, 16\}$ is the start hour. The sizes of the infrastructure network are shown in Table 6.1, the number of trains in Table 6.2. In all of our tests we used a time discretisation of one minute (which is a typical choice for TTP). We solved the Lagrangian relaxation (see Chapter 3) of our model using the proximal bundle method implemented in the ConicBundle [52] software package (except in the tests of the parallel bundle method, which use the newly developed and implemented algorithm). The coupling constraints (*i. e.*, all constraints of type (2.6)–(2.8)) have not been added statically to the model but have been separated if they were violated by the current primal aggregate solution (primal cutting planes are supported by ConicBundle, see [4, 51]). In most tests presented here we used the clique based model (TTP-clq) (see Section 2.5). We will discuss the algorithmic behaviour of the clique based and the configuration based model in Section 6.4.3. All algorithms have been implemented in C++ using the gcc 4.7.1 compiler and tested on an Intel Core i7 CPU with 12 GB RAM and eight 2.67 GHz cores.

6.2 Dynamic Graph Generation

In this section we present some tests for our Dynamic Graph Generation technique described in Chapter 4. For these tests we used the clique based model (TTP-clq), *i. e.*, we focused on the time expanded networks of the trains and not on the configuration networks (but see the discussion at the end of this section). Furthermore we considered two slight variations of our basic model: one, denoted by "a", in which passenger trains are allowed to stop only at the stations at which they must stop according to their predefined timetable, and a second variant, denoted by "b", in which passenger trains may stop at intermediate stations. The difference in the resulting model is that in variant "a" the

Figure 6.1: The South-West subnetwork of *Deutsche Bahn*. The short distance trains contained in the data are restricted to the black arcs. The grey arcs are only used by long distance and freight trains, *i. e.*, they are the extensions of the train runs of long distance and freight trains to the whole network.

instance	# freight	passenger # long-distance	passenger # short-distance	total # trains
RM-3	79	16	18	113
RM-4	82	19	21	122
RM-5	79	20	25	124
RM-6	81	23	27	131
RM-7	73	26	28	127
RM-8	68	26	27	121
RM-9	70	27	26	123
RM-10	71	28	27	126
RM-11	67	27	29	123
RM-12	67	28	29	124
RM-13	72	27	30	129
RM-14	71	27	28	126
RM-15	72	23	29	124
RM-16	71	24	27	122
SW-3	529	95	1372	1996
SW-4	545	112	1603	2260
SW-5	535	123	1821	2479
SW-6	495	135	1900	2530
SW-7	482	145	1877	2504
SW-8	464	147	1798	2409
SW-9	445	154	1759	2358
SW-10	446	152	1800	2398
SW-11	463	159	1919	2541
SW-12	477	161	2008	2646
SW-13	502	165	2024	2691
SW-14	492	157	1955	2604
SW-15	515	155	1880	2550
SW-16	538	145	1760	2443

Table 6.2: Number of trains in the instances.

intermediate stations have only run nodes, whereas in "b" they also have wait nodes. Each of those variants might be preferable in practice depending on whether stopping of passenger trains at intermediate stations should be forbidden or not.

We solved the Lagrangian relaxation of each instance with

(I) fully expanded networks,

(II) the simple corridor without shifting up of the corridor,

(III) the full dynamic graph generation with shifting up.

In the case of the fully expanded networks we counted the number of arcs at the beginning of the solution process (the networks are static so this number does not change during the solution process). Note that our model does not contain any additional "infeasibility" arcs for the case that some train may not have a feasible solution within the time horizon, but adding them would only increase the networks. For the tests with dynamic graph generation we counted the sizes at the end of the solution process, when all networks reached their maximal expansion (they grow dynamically during the solution process starting with only one path in the first evaluation).

The numbers of arcs in these networks are shown in Table 6.3 for the RM instances and in Table 6.4 for the SW instances. The number of arcs are divided into the three main train classes, long distance passenger trains, short distance passenger trains and freight trains, which have different characteristics. In addition, these tables show the reduction ratio $(II)/(III)$ of the number of arcs of the non-shifting and the shifting dynamic graph generation approach and the reduction ratio $(I)/(III)$ of the number of arcs of the full expanded network and the shifting dynamic graph generation approach.

As expected, dynamic graph generation reduces the actual sizes of the networks a lot. The reduction is best for short running trains (*i. e.* short distance passenger trains) because for these trains the "time-dimension" is large compared to the "space-dimension" (*i. e.* the length of the train route). However, because short distances trains have only short routes, the overall portion of arcs of short distance trains is rather small. The most significant class of trains are freight trains. These trains have a long route and contain many arcs and nodes (because freight trains always have the possibility to wait or to pass through a station in both model variants "a" and "b"). The difference between the shifted and non-shifted dynamic graph generation approach is largest for passenger trains and almost negligible for freight trains. The reason is that passenger trains contain forced waiting periods (because of the predefined timetable) whereas freight trains only have to reach their final destination as fast as possible, hence typically contain only few waiting periods. In model variant "a" the advantage of the shifting corridor is smaller (about a factor 1.5 for all trains in the SW instances) than for variant "b" (about a factor of 2). The explanation is that in variant "b" the passenger networks contain additionally wait nodes for the stopping possibilities at intermediate stations, and these wait nodes allow the shifting algorithm to construct a corridor that is closer to the active subgraph. The overall reduction of the number of arcs in the model when using the shifting dynamic graph generation approach is about a factor of 40. Note that a longer simulation horizon

instance	(I)	(II)	(III)	$\frac{(II)}{(III)}$	$\frac{(I)}{(III)}$
RM-3a	154793	16533	11010	1.50	14.06
RM-4a	192907	16523	11033	1.50	17.48
RM-5a	219543	16090	10349	1.55	21.21
RM-6a	246484	16727	11763	1.42	20.95
RM-7a	266942	16015	11889	1.35	22.45
RM-8a	267434	16544	12512	1.32	21.37
RM-9a	286796	17619	13283	1.33	21.59
RM-10a	290892	16244	12195	1.33	23.85
RM-11a	330618	17590	13507	1.30	24.48
RM-12a	326445	17637	13327	1.32	24.50
RM-13a	329622	18059	13607	1.33	24.22
RM-14a	302215	15114	11658	1.30	25.92
RM-15a	289704	15782	11643	1.36	24.88
RM-16a	259353	14005	10125	1.38	25.62
RM-3b	677930	64441	16610	3.88	40.81
RM-4b	845747	70632	19827	3.56	42.66
RM-5b	961107	68497	19353	3.54	49.66
RM-6b	1074423	68295	21336	3.20	50.36
RM-7b	1172344	70247	22293	3.15	52.59
RM-8b	1181841	73300	25037	2.93	47.20
RM-9b	1271574	75854	24778	3.06	51.32
RM-10b	1283755	74615	24427	3.05	52.55
RM-11b	1471465	80095	26974	2.97	54.55
RM-12b	1447698	75259	25895	2.91	55.91
RM-13b	1465919	80564	26678	3.02	54.95
RM-14b	1334995	68052	22773	2.99	58.62
RM-15b	1285643	69654	24152	2.88	53.23
RM-16b	1140229	62483	20955	2.98	54.41

(a) Long distance trains

instance	(I)	(II)	(III)	$\frac{(II)}{(III)}$	$\frac{(I)}{(III)}$
RM-3a	107820	5732	3081	1.86	35.00
RM-4a	171459	7642	3985	1.92	43.03
RM-5a	237283	9066	4886	1.86	48.56
RM-6a	265225	9791	5721	1.71	46.36
RM-7a	258704	9161	5640	1.62	45.87
RM-8a	222079	8226	5053	1.63	43.95
RM-9a	211401	8210	4876	1.68	43.36
RM-10a	210934	7716	4602	1.68	45.84
RM-11a	219244	8723	4824	1.81	45.45
RM-12a	234786	9404	4881	1.93	48.10
RM-13a	260118	10494	5501	1.91	47.29
RM-14a	276432	11001	5692	1.93	48.57
RM-15a	295769	10965	5648	1.94	52.37
RM-16a	289018	9467	5040	1.88	57.34
RM-3b	197474	11099	5692	1.95	34.69
RM-4b	317248	13919	7201	1.93	44.06
RM-5b	444594	16922	8420	2.01	52.80
RM-6b	508911	17435	8889	1.96	57.25
RM-7b	500712	16706	8804	1.90	56.87
RM-8b	420932	15332	8017	1.91	52.50
RM-9b	401502	14804	7542	1.96	53.24
RM-10b	399391	14648	7782	1.88	51.32
RM-11b	407846	15979	8813	1.81	46.28
RM-12b	442770	17000	8861	1.92	49.97
RM-13b	488727	18092	9632	1.88	50.74
RM-14b	523551	18733	9859	1.90	53.10
RM-15b	561558	17831	9511	1.87	59.04
RM-16b	548038	16280	8820	1.85	62.14

(b) Short distance trains

instance	(I)	(II)	(III)	$\frac{(II)}{(III)}$	$\frac{(I)}{(III)}$
RM-3a	4488437	188945	163339	1.16	27.48
RM-4a	4279795	178256	155432	1.15	27.53
RM-5a	4127336	171032	148496	1.15	27.79
RM-6a	3859079	166462	144045	1.16	26.79
RM-7a	3736845	161871	138932	1.17	26.90
RM-8a	3795128	156786	134289	1.17	28.26
RM-9a	3776954	149934	129149	1.16	29.24
RM-10a	3598488	140194	120407	1.16	29.89
RM-11a	3422322	144253	125611	1.15	27.25
RM-12a	3010701	113871	98230	1.16	30.65
RM-13a	2700088	116800	100413	1.16	26.89
RM-14a	2866489	126491	111028	1.14	25.82
RM-15a	3219261	149927	133790	1.12	24.06
RM-16a	3550654	175521	157565	1.11	22.53
RM-3b	4488437	198855	171996	1.16	26.10
RM-4b	4279795	187636	163435	1.15	26.19
RM-5b	4127336	183217	158828	1.15	25.99
RM-6b	3859079	181437	157349	1.15	24.53
RM-7b	3736845	178386	153251	1.16	24.38
RM-8b	3795128	172421	148455	1.16	25.56
RM-9b	3776954	164044	142758	1.15	26.46
RM-10b	3598488	154309	133202	1.16	27.02
RM-11b	3422322	155833	135969	1.15	25.17
RM-12b	3010701	125601	109276	1.15	27.55
RM-13b	2700088	127915	110458	1.16	24.44
RM-14b	2866489	137646	121118	1.14	23.67
RM-15b	3219261	162712	145512	1.12	22.12
RM-16b	3550654	185851	166756	1.11	21.29

(c) Freight trains

instance	(I)	(II)	(III)	$\frac{(II)}{(III)}$	$\frac{(I)}{(III)}$
RM-3a	4751050	211210	177430	1.19	26.78
RM-4a	4644161	202421	170450	1.19	27.25
RM-5a	4584162	196188	163731	1.20	28.00
RM-6a	4370788	192980	161529	1.19	27.06
RM-7a	4262491	187047	156461	1.20	27.24
RM-8a	4284641	181556	151854	1.20	28.22
RM-9a	4275151	175763	147308	1.19	29.02
RM-10a	4100314	164154	137204	1.20	29.88
RM-11a	3972184	170566	143942	1.18	27.60
RM-12a	3571932	140912	116438	1.21	30.68
RM-13a	3289828	145353	119521	1.22	27.53
RM-14a	3445136	152606	128378	1.19	26.84
RM-15a	3804734	176674	151081	1.17	25.18
RM-16a	4099025	198993	172730	1.15	23.73
RM-3b	5363841	274395	194298	1.41	27.61
RM-4b	5442790	272187	190463	1.43	28.58
RM-5b	5533037	268636	186601	1.44	29.65
RM-6b	5442413	267167	187574	1.42	29.01
RM-7b	5409901	265339	184348	1.44	29.35
RM-8b	5397901	261053	181509	1.44	29.74
RM-9b	5450030	254702	175078	1.45	31.13
RM-10b	5281634	243572	165411	1.47	31.93
RM-11b	5301633	251907	171756	1.47	30.87
RM-12b	4901169	217860	144032	1.51	34.03
RM-13b	4654734	226571	146768	1.54	31.71
RM-14b	4725035	224431	153750	1.46	30.73
RM-15b	5066462	250197	179175	1.40	28.28
RM-16b	5238921	264614	196531	1.35	26.66

(d) All trains

Table 6.3: Number of arcs in all graphs at the end of the solution process for RM instances

instance	(I)	(II)	(III)	$\frac{(II)}{(III)}$	$\frac{(I)}{(III)}$
SW-3a	822795	100295	46806	2.14	17.58
SW-4a	1290951	153588	66344	2.32	19.46
SW-5a	1883423	203653	86270	2.36	21.83
SW-6a	2400731	241395	100194	2.41	23.96
SW-7a	2527408	247489	100113	2.47	25.25
SW-8a	2578064	251007	102137	2.46	25.24
SW-9a	2691333	266758	106102	2.51	25.37
SW-10a	2679971	267310	104302	2.56	25.69
SW-11a	2764144	271703	104280	2.61	26.51
SW-12a	2755740	266815	101844	2.62	27.06
SW-13a	2797058	269463	105707	2.55	26.46
SW-14a	2752042	259875	105270	2.47	26.14
SW-15a	2775214	257317	105396	2.44	26.33
SW-16a	2721956	244291	103164	2.37	26.38
SW-3b	3449906	528871	215446	2.45	16.01
SW-4b	5456862	774920	289034	2.68	18.88
SW-5b	8014332	1027995	359192	2.86	22.31
SW-6b	10274682	1224973	418935	2.92	24.53
SW-7b	10879062	1255695	423178	2.97	25.71
SW-8b	11083461	1289864	434288	2.97	25.52
SW-9b	11606849	1350909	440825	3.06	26.33
SW-10b	11533257	1345717	440565	3.05	26.18
SW-11b	11934731	1374620	438994	3.13	27.19
SW-12b	11875565	1360759	444889	3.06	26.69
SW-13b	12065300	1369700	450271	3.04	26.80
SW-14b	11819615	1325428	443333	2.99	26.66
SW-15b	11921639	1294304	429128	3.02	27.78
SW-16b	11671216	1230013	403959	3.04	28.89

(a) Long distance trains

instance	(I)	(II)	(III)	$\frac{(II)}{(III)}$	$\frac{(I)}{(III)}$
SW-3a	5103300	199100	104270	1.91	48.94
SW-4a	8020270	251370	130158	1.93	61.62
SW-5a	11061357	296559	152556	1.94	72.51
SW-6a	12041403	304872	157873	1.93	76.27
SW-7a	11458603	290123	152343	1.90	75.22
SW-8a	10490112	284698	148526	1.92	70.63
SW-9a	10158403	280352	147074	1.91	69.07
SW-10a	10313094	286611	149304	1.92	69.07
SW-11a	10753436	302730	158856	1.91	67.69
SW-12a	11334169	316539	165185	1.92	68.62
SW-13a	11714004	326511	172520	1.89	67.90
SW-14a	11814837	324243	169561	1.91	69.68
SW-15a	12069255	314088	165006	1.90	73.14
SW-16a	12029039	288874	153188	1.89	78.52
SW-3b	9558204	477620	253272	1.89	37.74
SW-4b	15195471	607215	315909	1.92	48.10
SW-5b	21190841	719572	371037	1.94	57.11
SW-6b	23481237	743107	386890	1.92	60.69
SW-7b	22762321	725809	379237	1.91	60.02
SW-8b	21153588	719876	373789	1.93	56.59
SW-9b	20515725	711107	371019	1.92	55.30
SW-10b	20767422	721984	372558	1.94	55.74
SW-11b	21519074	757842	393035	1.93	54.75
SW-12b	22613054	794096	409511	1.94	55.22
SW-13b	23346832	813624	424679	1.92	54.98
SW-14b	23655000	811215	418081	1.94	56.58
SW-15b	24119589	781724	406316	1.92	59.36
SW-16b	24065501	720860	375764	1.92	64.04

(b) Short distance trains

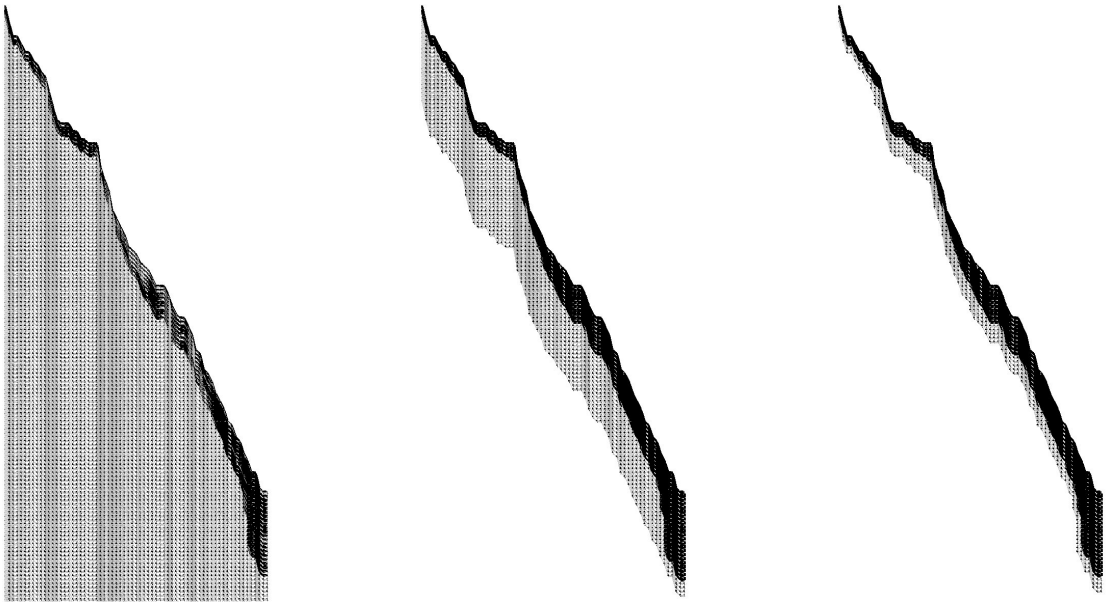
instance	(I)	(II)	(III)	$\frac{(II)}{(III)}$	$\frac{(I)}{(III)}$
SW-3a	20893942	511384	506109	1.01	41.28
SW-4a	21019203	518565	513980	1.01	40.89
SW-5a	20673212	525947	522309	1.01	39.58
SW-6a	20189047	534321	528306	1.01	38.21
SW-7a	20053543	519934	514839	1.01	38.95
SW-8a	19568062	503825	498250	1.01	39.27
SW-9a	18964440	477209	472477	1.01	40.14
SW-10a	18248704	435786	430704	1.01	42.37
SW-11a	17180833	410875	406128	1.01	42.30
SW-12a	15884968	386139	383214	1.01	41.45
SW-13a	15109297	382914	380199	1.01	39.74
SW-14a	14530858	383089	378783	1.01	38.36
SW-15a	14714232	402742	397714	1.01	37.00
SW-16a	15623100	433290	431322	1.00	36.22
SW-3b	20893942	550979	544914	1.01	38.34
SW-4b	21019203	579475	571856	1.01	36.76
SW-5b	20673212	609727	602392	1.01	34.32
SW-6b	20189047	602521	596766	1.01	33.83
SW-7b	20053543	596814	588693	1.01	34.06
SW-8b	19568062	581650	573891	1.01	34.10
SW-9b	18964440	545444	537362	1.02	35.29
SW-10b	18248704	490876	485388	1.01	37.60
SW-11b	17180833	461795	456497	1.01	37.64
SW-12b	15884968	445109	439397	1.01	36.15
SW-13b	15109297	451614	446678	1.01	33.83
SW-14b	14530858	442459	437572	1.01	33.21
SW-15b	14714232	470107	462747	1.02	31.80
SW-16b	15623100	506430	499455	1.01	31.28

(c) Freight trains

instance	(I)	(II)	(III)	$\frac{(II)}{(III)}$	$\frac{(I)}{(III)}$
SW-3a	26820037	810779	657185	1.23	40.81
SW-4a	30330424	923523	710482	1.30	42.69
SW-5a	33617992	1026159	761135	1.35	44.17
SW-6a	34631181	1080588	786373	1.37	44.04
SW-7a	34039554	1057546	767295	1.38	44.36
SW-8a	32636238	1039530	748913	1.39	43.58
SW-9a	31814176	1024319	725653	1.41	43.84
SW-10a	31241769	989707	684310	1.45	45.65
SW-11a	30698413	985308	669264	1.47	45.87
SW-12a	29974877	969493	650243	1.49	46.10
SW-13a	29620359	978888	658426	1.49	44.99
SW-14a	29097737	967207	653614	1.48	44.52
SW-15a	29558701	974147	668116	1.46	44.24
SW-16a	30374095	966455	687674	1.41	44.17
SW-3b	33902052	1557470	1013632	1.54	33.45
SW-4b	41671536	1961610	1176799	1.67	35.41
SW-5b	49878385	2357294	1332621	1.77	37.43
SW-6b	53944966	2570601	1402591	1.83	38.46
SW-7b	53694926	2578318	1391108	1.85	38.60
SW-8b	51805111	2591390	1381968	1.88	37.49
SW-9b	51087014	2607460	1349206	1.93	37.86
SW-10b	50549383	2558577	1298511	1.97	38.93
SW-11b	50634638	2594257	1288526	2.01	39.30
SW-12b	50373587	2599964	1293797	2.01	38.93
SW-13b	50521429	2634938	1321628	1.99	38.23
SW-14b	50005473	2579102	1298986	1.99	38.50
SW-15b	50755460	2546135	1298191	1.96	39.10
SW-16b	51359817	2457303	1279178	1.92	40.15

(d) All trains

Table 6.4: Number of arcs in all graphs at the end of the solution process for SW instances



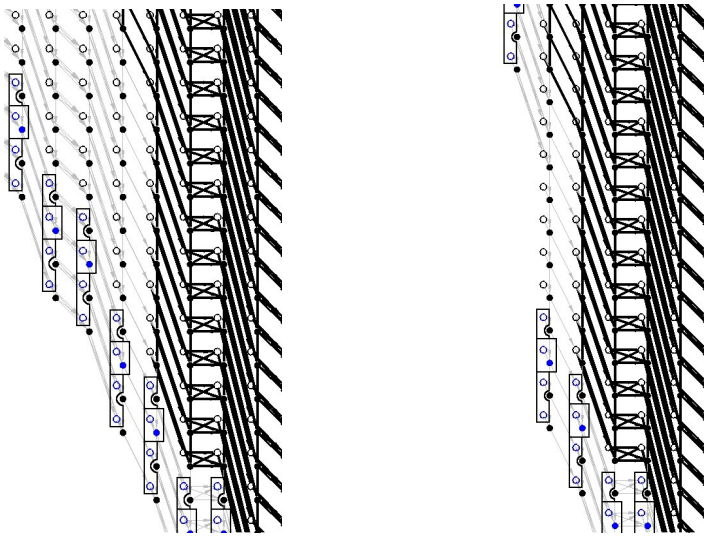
(I) Expanded graph (II) Simple corridor (III) Improved corridor

Figure 6.2: An example of a freight train graph. The thicker arcs at the top show the graph G^{act}. The freight train has a waiting period near the end which causes the simple corridor to contain more arcs than the improved corridor.

would lead to an even larger factor. This huge reduction gives a significant improvement for large scale practical instances.

In order to illustrate the effect of dynamic graph generation, we visualised the graphs of some trains. Figure 6.2 shows the respective final network of one particular freight train at the end of the solution process in each of the three cases. Nodes and arcs too early to be reached from $(\hat{u}, 1)$ are neither generated nor counted in any of these graphs. The dark arcs at the top of the graph are the arcs that belong to G^{act}, *i. e.*, these are contained in some generated shortest path. The fully expanded network (I) contains a large number of arcs for later time steps that have never been used and most of them are not generated in (II) or (III). The train has a waiting period near the end of its route. Waiting periods may lead to relatively large corridors when using the simple approach, because the simple corridor contains the *fastest* paths. Thus the earliest possible starting time when running at full speed and arriving below the last G^{act}-arc at the final station causes the corridor to be relatively far away from G^{act}. In contrast, the improved corridor (III) uses potential wait arcs to shift the corridor to earlier time steps which keeps it close to G^{act} at all time steps.

Figure 6.3 gives a close up view of the waiting period near the end of the train run for both corridors. The nodes within the "rectangles" indicate the sets $V^{[u]}_{t_{[u]}}$. As one can see, the simple corridor has to keep those sets so that a continuation along the fastest route is possible. In contrast, the improved corridor may shift some of those sets to earlier time



(I) Simple corridor (II) Improved corridor

Figure 6.3: Part of simple and improved corridor. The nodes within the "rectangles" correspond to sets $V^{[u]}_{t_{[u]}}$. The improved corridor uses wait arcs to reduce the size of the corridor at earlier stations.

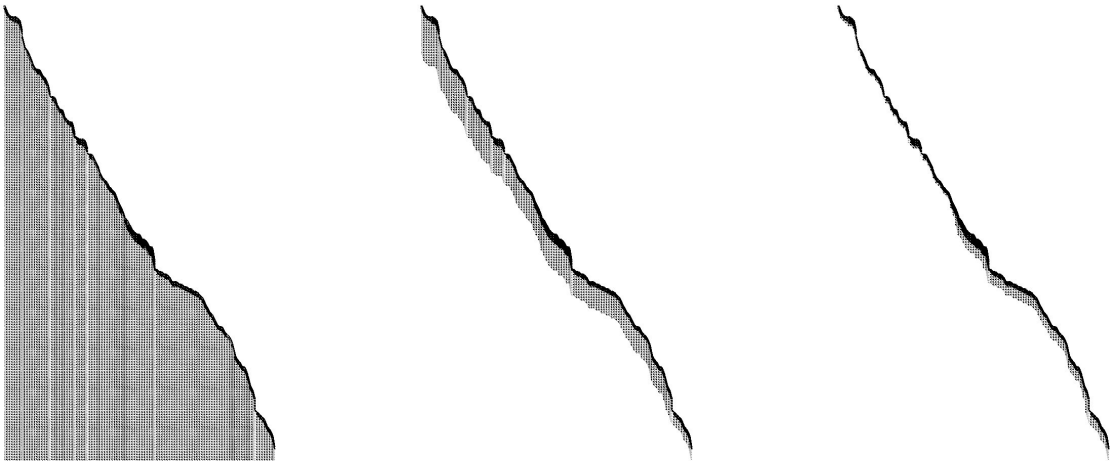
steps if a continuation along wait arcs is possible.

Another example is a passenger train shown in Figure 6.4. Passenger trains often have forced waiting periods because of timetable restrictions (the train must not leave a station before some time step). Because of this, the fastest possible route may in general be faster than the fastest route along the earliest time steps (which has some forced waiting periods). Therefore, generically, the simple corridor (II) deviates a lot from G^{act} on the first clone nodes visited. Again, the improved corridor (III) exploits intermediate wait nodes to keep the corridor close to G^{act} at all stations.

We close this section with a remark about configuration networks. Configuration networks do not fit in our constructive framework for dynamic graph generation (see Section 4.5). However, the general concept works for them, too, only the construction of the current subnetwork is different (in fact, much simpler). Configuration networks are even larger than train graphs: Consider a track with 100 trains in six hours (such tracks exist, *e. g.* on the "Rhein-Main-Schiene"). For each time step the network contains about (see Section 2.6)

- one configuration arc and one wait arc for each train run (particularly freight trains contain often four runs, one for each possible running behaviour),
- one headway arc for each ordered pair of trains.

This means over a horizon of six hours and a discretisation of one minute (360 time steps) a single network contains about $360 \cdot 100 \cdot 99 = 3564000$ arcs. Note that the infrastructure network contains thousands of tracks, so that fully expanded configuration networks would



(I) Expanded graph (II) Simple corridor (III) Improved corridor

Figure 6.4: An example of a long distance train graph. The thicker arcs at the top show the graph G^{act}. Passenger trains often have forced waiting periods due to timetable restrictions. The improved corridor is designed to exploit such waiting periods.

lead to intractable models. Because of this, we did not test fully expanded configuration networks but always generated them dynamically.

6.3 Asynchronous Parallel Bundle Method

In this section we present first numerical results comparing the (Extended) Parallel Bundle Method presented in Chapter 5 with a standard proximal bundle algorithm. Note that it is not the intent of this thesis to provide an extensive numerical study. Instead the focus was on theoretic foundations of the parallel bundle method and the purpose of the tests is to show that there are problems where the application of the parallel bundle method may give advantages over the classical bundle method and may therefore be a valid alternative in future applications, in particular in train timetabling problems.

The large scale real world instances of the TTP require fine grained models like those described in Chapter 2 and often more advanced techniques like primal cutting planes and primal column generation in conjunction with Dynamic Graph Generation (see Chapter 4) in order to be efficiently treatable. The current prototype implementation of the parallel bundle algorithm cannot handle these cases, yet, and therefore we restricted to randomly generated test instances with static networks and no separation of constraints. The construction of those instances is described in the next section. Afterwards in Section 6.3.2 we present the results of our tests.

6.3.1 Test Instances

We used simple randomly generated test instances for the *Train Timetabling Problem*, see Chapter 2 for a detailed description. In a real world TTP problem there are often certain local areas with many local short distance trains (*e. g.*, around big cities) and a number of long distance and freight trains running through the whole network and coupling those local areas. In order to mimic this structure, we generated in our tests a random network as a subset of the two dimensional square grid graph which is further divided into several (almost disjoint) subsquares. Those subsquares represent the local areas and each subsquare is filled afterwards with a certain number of local trains with random routes within that subsquare and random start time within a certain time window. Analogously a certain number of global trains is generated which have random routes within the whole network coupling the local areas. Each train has a running time of either 1 or 2 per arc chosen randomly, all nodes in the network have a random capacity of either 1 or 2 and all headway times are assumed to be one minute.

As stated before, we are not yet able to use Dynamic Graph Generation combined with the parallel bundle method. However, when using static networks we have to ensure that the time horizon in each network is large enough so that a feasible solution exists. In our case we simply expanded each network up to twice its minimal running time from its start to its end node.

The node set of our test instances is constructed as follows. Let $n_l, d_l \in \mathbb{N}$ bet two numbers. The node set consists of $n_l \times n_l$ subsquares $U^I_{i,j} = \{(u_1, u_2) \colon u_1 \in \{(i-1)d_l, \ldots, id_l\}, u_2 \in \{(j-1)d_l, \ldots, jd_l\}$ and the full node set $U^I = \bigcup_{i,j=1}^{n_l} U^I_{i,j}$. The arc set A^I is generated by inserting randomly a number of routes from a node of one border of the square node set to a node of the opposite site. The detailed approach can be seen in Algorithm 6.1.

After the arc set is constructed a set of trains is generated. For this a random border node is chosen and from there a random path within the arc set until another border node is reached. If a loop is created or the generated train route is too short, the route is thrown away and the procedure is restarted. Algorithm 6.2 is called n_{local} times for each local area and n_{global} times for the whole network to generate local and global trains, respectively.

Figure 6.5 shows an example test instance for 3×3 subsquares. In our test instances we chose $n_l = 20$, $d_l = 4$, $n_{\text{local}} = 60$, and $n_{\text{global}} \in \{40, 60, 80, 100, 120\}$ to generate 16 instances per choice of n_{global}.

6.3.2 Tests

We ran 80 test instances with 4×4 subsquares, each of size 20×20, forming a large square of size 80×80 and 60 local trains per subsquare starting randomly distributed within three hours. The number of global trains that couple the local areas is increased from 40 to 120 trains in steps of 20 and they are started within three hours, too.

Both algorithms, the classical bundle method and the parallel bundle method, have been implemented in C++. The quadratic subproblem (see (5.3) in Section 5.2) has been

Algorithm 6.1: Generation of infrastructure arcs

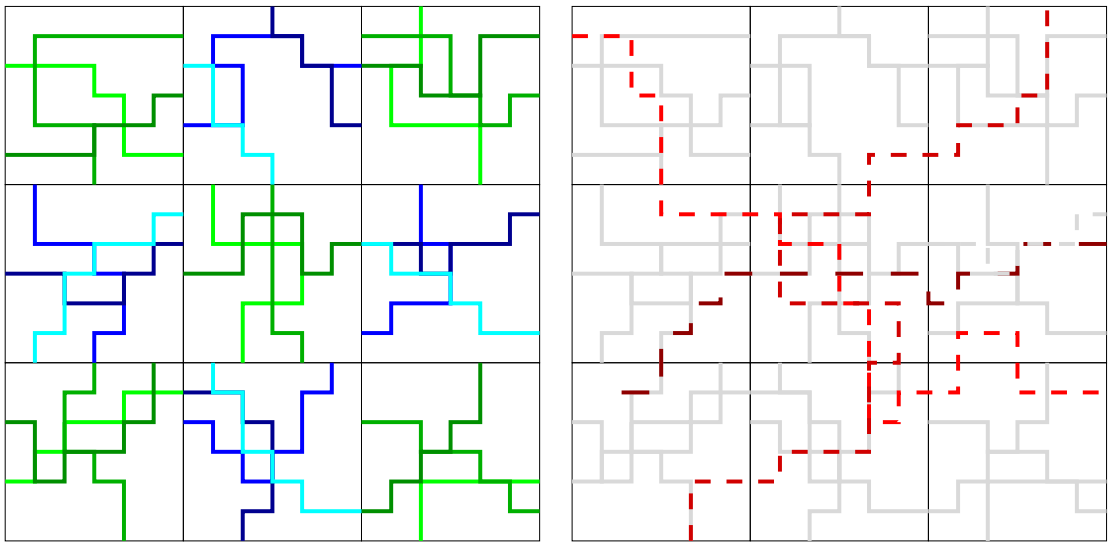
Input : n
Output: $A^I \subseteq (U^I)^2$
for $k := 1$ *to* n **do**
  Choose a direction $d \in \{-e_1, e_1, -e_2, e_2\}$ randomly.
  Choose $u_0 \in U^I$ with $d^T u_0$ minimal `// i. e., u_0 is a border node`
  $i \leftarrow 0$.
  while $d^T u_i$ *not maximal* `// i. e., u_i has not reached the opposite border` **do**
    Choose $u_{i+1} \in U$ with `// i. e., not in backward direction.`

$$\|u_{i+1} - u_i\| = 1,$$
$$(u_{i+1} - u_i)^T d \geq 0,$$
$$i = 0 \vee u_{i-1} \neq u_{i+1}$$

    $A^I \leftarrow A^I \cup \{(u_i, u_{i+1})\}$
    $i \leftarrow i + 1$.

Algorithm 6.2: Generation of trains

Input : Subsquare $G' = (U', A')$ with size n.
Output: Train route $(u_1, \ldots, u_k)$
Choose random border node $u_0 \in U'$.
$i \leftarrow 0$
repeat
  $i \leftarrow i + 1$
  repeat
    Choose u_i as random neighbour of u_{i-1}
  until $u_i \notin \{u_0, \ldots, u_{i-1}\}$
until u_i *border node*

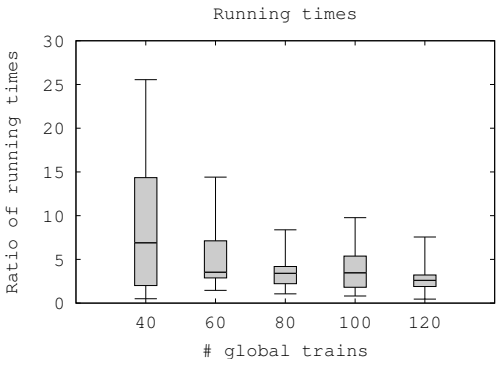


(a) Short distance trains (b) Long distance trains

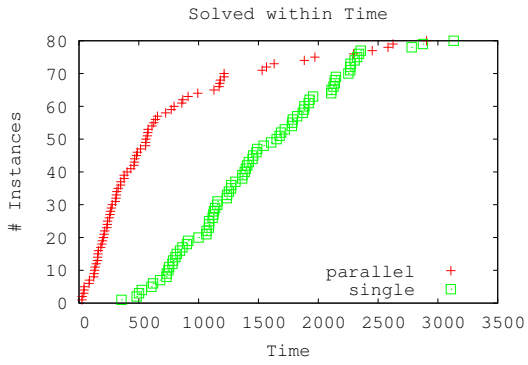
Figure 6.5: Test instance for Parallel Bundle Algorithm. The two figures show the routes and some example local short distances resp. global long distance trains.

solved with CPLEX 12.1 [56]. For better comparability we used the same bundle code for the classical bundle method and each subprocess bundle method (*i. e.*, we did not use CONICBUNDLE for the classical bundle method). We solved all instances up to a final termination precision of 10^{-4}. In a first test we solved all instances with the classical bundle method on one core and with the parallel bundle method on one core with exactly one subprocess. In the second test we ran the tests with the classical bundle method where the subproblem evaluations were executed in parallel on 4 cores and the parallel bundle method with up to 4 parallel processes. Figure 6.6 shows the results for the tests with one core. Denoting the running times by t_{parallel} for the parallel bundle method and t_{single} for the classical bundle method, diagram (a) shows the minimum, the maximum, the 25%, 75% quantiles and the median of the relative running times $t_{\text{single}}/t_{\text{parallel}}$ over all instances with a certain number of global trains. Diagram (b) shows the number of instances solved up to a certain time for both algorithms. Figure 6.7 shows the analog results for the tests on four cores.

The diagrams show that the parallel bundle method solves most instances within a shorter time than the classical bundle method on one as well as on four cores. Furthermore, as the number of global trains is increased the advantage of the parallel bundle method decreases. This matches expectations, because the classical bundle method has to evaluate all subproblems equally often whereas the parallel bundle method may take advantage of the decoupled structure of the problem with some easy and some more difficult areas. Therefore the parallel bundle method may evaluate certain subproblems less often while focusing on the more difficult ones. When the number of global trains is increased, coupling between the subproblems is increased as well, and hence most subproblems

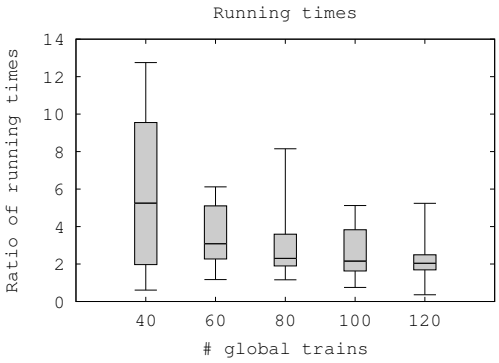


(a) minimum, maximum, 25%, 75% quantiles and median of $t_{\text{single}}/t_{\text{parallel}}$

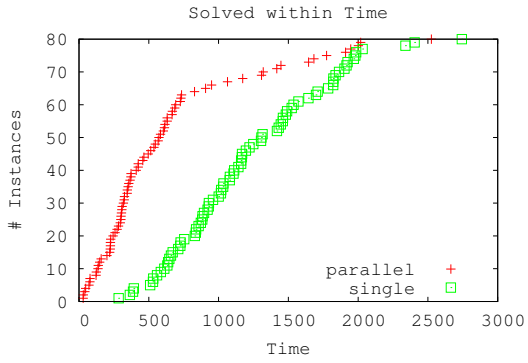


(b) Performance profile

Figure 6.6: Running times for 80 test instances on 1 core.



(a) minimum, maximum, 25%, 75% quantiles and median of $t_{\text{single}}/t_{\text{parallel}}$



(b) Performance profile

Figure 6.7: Running times for 80 test instances on 4 cores.

become similarly difficult. In this case the parallel bundle method looses its advantage.

It may come as a surprise, however, that the improvement of using four cores instead of one is not evident, not only for the parallel but also for the classical bundle method. A possible explanation might be that the parallel evaluation of a large number of combinatorial subproblems on multicore machines with common main memory seems to cause significant problems in accessing memory (cache) and the speedup in pure computing power is lost due to competing memory accesses. Therefore much better implementations of the algorithms are required in order to properly exploit the properties of modern hardware. One important possible step in this direction could be a cache optimised implementation of Dynamic Graph Generation and an extension of the parallel bundle algorithm to support this technique: the small size of the stored networks allow for much more efficient access to the shared memory potentially giving a big performance boost on parallel hardware.

Still, the numerical results indicate that the proposed parallel bundle approach has the potential for developing into a useful alternative for large scale applications with appropriate structure.

6.4 Train Timetabling Problem

In this section we give a few results concerning our real world instances. All instances have been solved using our Dynamic Graph Generation techniques. In fact, without this technique they seemed rather intractable, having huge memory requirements and also large running times for the algorithms working on the large fully expanded networks.

The basic solution approach is to solve the Lagrangian relaxation (see Section 3.3) of the TTP. This gives approximate dual and primal fractional solutions. The fractional solutions are then used to guide a simple rounding heuristic in order to produce integer solutions. We describe the rounding heuristic in Section 6.4.1 and present the numerical results in Section 6.4.2.

6.4.1 Rounding Heuristic

Currently we used a rather simple rounding heuristic that works as follows. In an iterative algorithm we first solve the Lagrangian relaxation of the model approximately. One or more of these trains are fixed afterwards and the next iteration is started, computing a new Lagrangian relaxation w. r. t. the fixed trains. The heuristic first fixes long distance trains, then short distance trains and finally freight trains (the precedence is a general rule given by *Deutsche Bahn*). These steps are repeated until all trains are fixed.

In detail, in each iteration the rounding procedure works as follows. The basic operation is the computation of an integer solution which is close to the fractional solution. For this we use the negative fractional value of each travel arc as coefficient in the objective function (wait arcs get the coefficient 0). In this computation we ignore those arcs that are blocked by previously fixed train runs (*i. e.*, arcs that would lead to a violated constraint with the already fixed trains), so that the resulting path is not in conflict with the fixed trains. The computation of a path is shown Algorithm 6.3.

Algorithm 6.3: FracPath

Input :
- current train graph G_r^{cur},
- fractional solution $\bar{x}_r \colon A_r^T \to [0,1]$,
- minimal arc flow $\delta > 0$,
- set of blocked arcs B

Output: A path $P^* \in \mathcal{P}_r^{\text{cur}}$

Set objective function

$$\xi_r(a) := \begin{cases} 0, & \text{if } a \notin B \text{ and } a \text{ is a wait arc,} \\ -\bar{x}_r(a), & \text{if } a \notin B \text{ and } a \text{ is a travel arc and } \bar{x}_r(a) \geq \delta, \\ \infty, & \text{otherwise.} \end{cases}$$

Determine $P^* \in \operatorname{Argmin}\{\xi_r(P) \colon P \in \mathcal{P}_r^{\text{cur}}\}$

return P^*

The rounding procedure first selects the class of candidate trains R^{cand} to be considered. If there is at least one unfixed long distance train, the unfixed long distance trains are selected as candidates. Otherwise, if there are unfixed short distance trains, the unfixed short distance trains are selected. If all passenger trains are fixed, then the unfixed freight trains are selected. For each candidate train $r \in R^{\text{cand}}$ we compute a path P_r with Algorithm 6.3 and determine the smallest fractional value of a travel arc on this path. The maximum of these values over all candidates is determined afterwards and denoted by $\varepsilon_{\text{accept}}$.

Finally we try iteratively each candidate train r and compute a path $\hat{P}_r$ for this train w. r. t. all previously fixed trains. If the smallest fractional value of a travel arc is at least $\max\{\varepsilon_{\text{accept}} - 0.1, 0\}$ then this path is fixed and the set of blocked arcs is updated. Then the next train is tried. Note that the computed path may differ from the path determined at the beginning of this iteration because in the meantime additional arcs may get blocked because of newly fixed trains. Furthermore, at least one train will be fixed in each iteration (there is at least one train whose fractional value is large enough). The rounding heuristic is shown in Algorithm 6.4.

6.4.2 Results for Real World Instances

The results for all instances are shown in Table 6.5, the running times in Table 6.6. All instances could be solved within 15 hours, most instances in less than 12 hours. Note that the solution process is very unstable and the number of trains fixed in each iteration varies a lot.

The objective function we used (see Section 2.7) is artificial and has no direct interpretation for practice. Instead, it was used to model the general goals of minimising delays of passenger trains and the arrival time of freight trains. Therefore we use the maximal delay of a passenger train at some of its stopping stations and the arrival time

Algorithm 6.4: SolveTTP

Input : A TTP instance
Output: A solution
define $R^{\text{ld}} := \{r \in R\colon \theta(r) \in \Theta^{\text{ld}}\}$ `// long distance trains`
define $R^{\text{sd}} := \{r \in R\colon \theta(r) \in \Theta^{\text{sd}}\}$ `// short distance trains`
define $R^{\text{fr}} := \{r \in R\colon \theta(r) \in \Theta^{f}\}$ `// freight trains`
set $R^{\text{fixed}} \leftarrow \emptyset$ `// already fixed trains`
set $B \leftarrow \emptyset$ `// blocked arcs`
while $R^{fixed} \neq R$ **do**
  Solve relaxation $\rightsquigarrow$ fractional solutions $\bar{x}_r \in \operatorname{conv} \mathcal{X}_r$, $r \in R$
  `// determine candidates, start with long distance, short distance, freight`
  if $R^{ld} \not\subseteq R^{fixed}$ **then**
    set $R^{\text{cand}} \leftarrow R^{\text{ld}} \setminus R^{\text{fixed}}$
  else if $R^{sd} \not\subseteq R^{fixed}$ **then**
    set $R^{\text{cand}} \leftarrow R^{\text{sd}} \setminus R^{\text{fixed}}$
  else
    set $R^{\text{cand}} \leftarrow R^{\text{fr}} \setminus R^{\text{fixed}}$
  Determine best fractional paths $P_r \leftarrow \textsc{FracPath}(G_r^{\text{cur}}, \bar{x}_r, 0, B)$, $r \in R^{\text{cand}}$
  Compute accept-bound

$$\varepsilon_{\text{accept}} \leftarrow \max\left\{\min\{\bar{x}_{r,a}\colon a \in A_r^T(P_r), a \text{ is travel arc}\}\colon r \in R^{\text{cand}}\right\}$$

  foreach $r \in R^{cand}$ **do**
    Determine path $\hat{P}_r \leftarrow \textsc{FracPath}(G_r^{\text{cur}}, \bar{x}_r, \max\{0, \varepsilon_{\text{cand}} - 0.1\}, B)$
    Compute $\varepsilon_r \leftarrow \min\{\bar{x}_{r,a}\colon a \in A_r^T(P_r), a \text{ is travel arc}\}$
    if $\varepsilon_r \geq \varepsilon_{accept} - 0.1$ **then**
      Fix path $\hat{P}_r$
      Set $R^{\text{fixed}} \leftarrow R^{\text{fixed}} \cup \{r\}$
      Update blocked arcs $B \leftarrow B \cup \{\text{arcs blocked by } \hat{P}_r\}$

instance	long distance delay				short distance delay				freight savings		
	max	avg	dev	good	max	avg	dev	good	max	avg	dev
RM-3	48	19.88	9.66	1.000	390	53.53	58.34	1.000	14868	4715.01	4018.98
RM-4	48	20.80	10.24	1.000	354	59.72	67.51	1.000	14820	4317.13	3884.41
RM-5	90	27.79	21.87	1.000	354	51.62	61.59	1.000	14670	4317.27	3908.03
RM-6	78	28.24	21.06	1.000	354	48.60	57.80	1.000	14739	4172.73	3859.60
RM-7	138	35.08	37.64	1.000	354	50.19	57.05	1.000	14739	4359.38	3391.60
RM-8	78	21.27	19.21	1.000	354	57.54	63.13	1.000	14739	4707.16	3206.27
RM-9	78	24.67	21.99	1.000	312	47.18	50.29	1.000	16659	4512.84	3785.04
RM-10	66	21.00	18.73	1.000	252	49.84	49.44	1.000	18030	4431.14	4349.63
RM-11	72	20.73	18.34	1.000	252	51.04	47.83	1.000	16620	4416.81	4286.39
RM-12	66	21.60	17.01	1.000	192	36.43	30.29	1.000	15552	4219.00	4050.96
RM-13	78	24.00	19.64	1.000	192	36.90	29.45	1.000	14742	3841.28	3804.03
RM-14	66	27.00	21.42	1.000	132	36.23	26.14	1.000	11982	3780.70	3242.38
RM-15	138	27.43	33.08	1.000	192	37.86	29.95	1.000	14276	4098.14	3322.46
RM-16	30	15.60	6.68	1.000	150	35.89	28.66	1.000	14276	4155.90	3364.59
SW-3	582	54.41	93.66	1.000	1980	95.66	145.34	0.994	32698	3551.87	5258.88
SW-4	582	81.29	112.27	1.000	6990	102.45	226.02	0.995	32698	3477.12	5319.26
SW-5	582	107.22	136.38	1.000	4176	109.40	221.51	0.989	28678	3448.44	5258.95
SW-6	582	97.92	122.70	1.000	5322	107.20	227.96	0.994	30427	3766.39	5485.37
SW-7	642	114.60	133.47	1.000	6984	105.52	238.93	0.995	29587	3904.30	5528.78
SW-8	1836	161.44	275.94	0.993	4728	110.85	242.73	0.992	29107	3883.17	5389.44
SW-9	1818	175.16	332.93	0.981	7080	120.14	349.29	0.994	24729	3823.91	5170.35
SW-10	918	171.09	212.12	1.000	5520	118.50	308.40	0.993	23460	3613.02	5024.84
SW-11	498	105.48	120.94	1.000	5520	112.49	281.27	0.994	45830	3481.39	5519.35
SW-12	462	88.48	106.81	1.000	17880	146.24	737.96	0.993	43310	3232.25	5373.18
SW-13	618	111.72	123.27	1.000	7740	152.86	515.57	0.987	40670	2908.06	5009.78
SW-14	690	107.92	117.79	1.000	8334	149.01	491.08	0.988	36410	2862.94	4830.80
SW-15	648	125.15	150.25	1.000	12468	143.33	510.72	0.991	35090	2667.23	4515.44
SW-16	618	128.32	138.34	1.000	12888	144.52	510.99	0.986	33110	2559.16	4222.57

Table 6.5: TTP results. The table shows the maximal and average delay and the standard deviation of delays of the long distance and short distance passenger trains as well as the maximal and average savings and standard deviations of savings of the arrival time at the final station of freight trains. Columns 5 and 9 show the relative amount of "good" passenger trains, *i. e.*, those trains that have a delay of less than 15 minutes.

of a freight train at the final station compared with the arrival time in the original data as a measure of quality. Table 6.5 shows the maximal and the average delay of passenger trains (in seconds) and the standard deviation of the delay. Furthermore the portion of "good" trains is shown, these are trains that have a delay of less than 15 minutes. For freight trains we do not show the delay but the difference of the original arrival time minus the arrival time in the solution. We show again the maximal and the average saving (in seconds) as well as the standard deviation.

The results for the RM instances are very good, with almost no delay for all passenger trains. Note that a delay of about one minute is expected because we use a time discretisation of one minute. In average the arrival time of a freight train could be reduced by more than one hour. For the larger instances (SW) the computed timetables are still very good. However, some trains suffered a relatively large delay (about 30 minutes for long distance trains and up to several hours for short distance trains). These trains should be considered failed in the current solutions. The reason why short distance trains have

instance	running time in seconds	instance	running time in seconds
RM-3	1758	SW-3	14741
RM-4	4085	SW-4	38187
RM-5	7762	SW-5	53714
RM-6	3257	SW-6	52783
RM-7	6260	SW-7	42460
RM-8	2729	SW-8	43838
RM-9	3440	SW-9	35329
RM-10	3647	SW-10	22265
RM-11	5587	SW-11	45472
RM-12	1330	SW-12	18765
RM-13	1930	SW-13	37081
RM-14	1880	SW-14	27210
RM-15	5183	SW-15	28160
RM-16	12900	SW-16	22222

Table 6.6: Running times of the solution algorithm

this big delay is that long distance trains are strongly preferred by the objective function. But note that the number of bad trains (which have a delay of more than 15 minutes) is still very small: usually more than 99% of all trains are "good" with only a small delay, which means that only about 20 out of 2000 could not be planned well. The savings for the freight trains, however, are very large, about one hour on average.

Also note that the results for the early instances starting at 3:00 and 4:00 are much better than for the instances starting around noon. The reason is that from 10:00 to 16:00 much more short distance passenger trains run in the network than early in the morning (but less freight trains).

The large freight savings indicate a general problem of the current time-expanded models (and of other models, as well): the fractional solutions computed by the Lagrangian relaxation approach tend to overestimate the capacity of the network. Consequently, freight trains run relatively early. However, this overestimation leads to conflicts in the rounding process and trains have to be scheduled much later than estimated by the fractional solution.

6.4.3 Clique Inequalities vs. Configuration Networks

In this last section we want to discuss the two possible models (TTP-clq) (see Section 2.5) and (TTP-cfg) (see Section 2.6). Borndörfer and Schlechte [9] showed that formulations based on configuration networks lead to stronger relaxations than those based on clique constraints. The reason is that the conflict graphs are usually quite complex. Separating clique constraints requires the computation of violated (maximal) cliques in the conflict graph. This problem is quite hard in general, so that, in practice, one can only separate very few clique constraints, *e. g.* only conflicts between each two trains (this is what we

do, too).

However, in our experiments it turned out that the clique based models behave *algorithmically* much better than configuration based models if used in a Lagrangian relaxation approach with first order methods. We want to demonstrate this on a tiny example: Consider a network with only one arc and two trains A and B. Both trains start at time index one and all headway times are 10 time steps. Train A is preferred, *e. g.* A may be a long-distance train and B a freight train. Consequently, in the optimal solution A will run immediately at time index 1 and B will follow at time index 11. The clique based model reads (we only show the parts that are relevant for our illustration)

$$\begin{aligned}
\text{minimise} \quad & \sum_{r\in\{A,B\}} \langle w_r, x_r\rangle \\
\text{subject to} \quad & \sum_{r\in\{A,B\}} \sum_{t=1}^{\bar{t}} x_{r,t} \leq 1, && \bar{t} \in \{1,\dots,10\}, && (6.1)\\
& x_r \in \mathcal{X}_r, && r \in \{A,B\}. && (6.2)
\end{aligned}$$

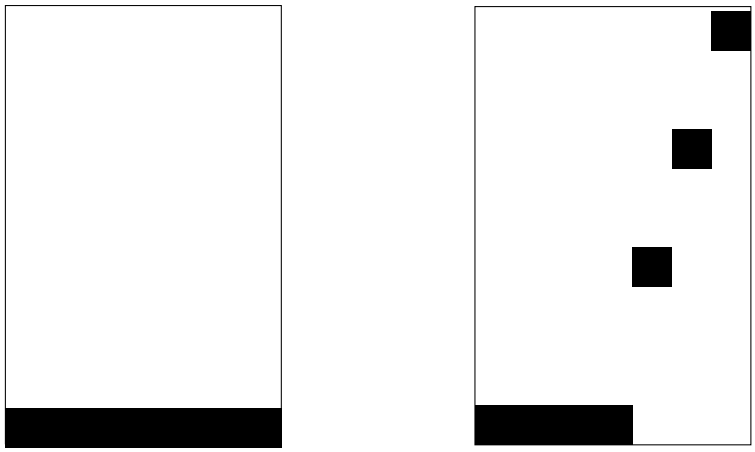
The variable $x_{r,t}$, $r \in \{A,B\}$, $t \in \{1,\dots,11\}$, corresponds to the run arc of train r starting at time t (there are also wait arcs in x_r but we neglect them here). Condition (6.2) states that x_r corresponds to a path in the network as usual. The constraints (6.1) are the clique constraints that state that only one of the run arcs in the time steps $\{\bar{t}-9,\dots,\bar{t}\}$, $\bar{t} \in \{1,\dots,10\}$, may be used in a solution (in fact, only the constraint for $\bar{t}=10$ is strictly necessary for this tiny example).

In the first iteration both trains run at time index one. The clique constraints will be violated and the costs of the first ten arcs will be increased (the costs of the earlier arcs are increased more quickly because these arcs are contained in more cliques). If the costs are increased sufficiently the freight train B will use a later run arc. The clique constraints, in particular the constraint for $\bar{t}=10$, increases the costs of *all* arcs with $t \in \{1,\dots,10\}$ at the same time, so the run of train B will be moved very quickly to the later time steps until it reaches time step 11. Figure 6.8 shows the development of the fractional solutions of both trains. After few iterations the fractional solution converges to the optimal solution.

Now consider the configuration based model. It reads

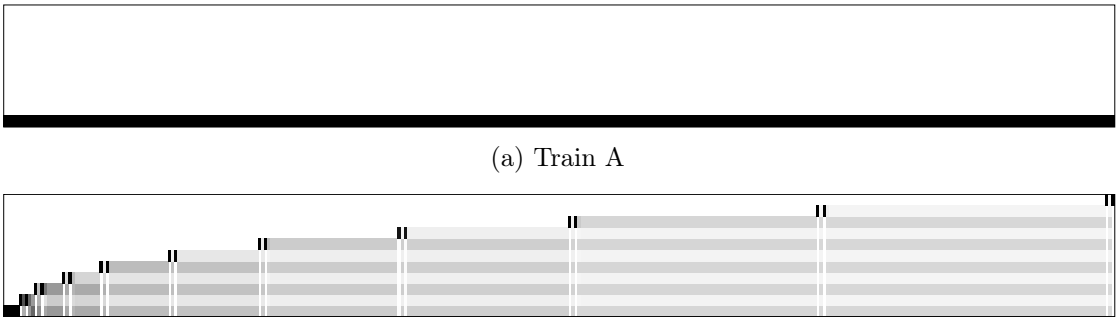
$$\begin{aligned}
\text{minimise} \quad & \sum_{r\in\{A,B\}} \langle w_r, x_r\rangle \\
\text{subject to} \quad & x_{r,t} = \bar{x}_{r,t}, && r \in \{A,B\}, t \in \{1,\dots,11\}, && (6.3)\\
& x_r \in \mathcal{X}_r, && r \in \{A,B\}, && \\
& \bar{x} \in \mathcal{P}, && && (6.4)
\end{aligned}$$

where $\mathcal{P}$ is the set of paths in the configuration network (according to the definition in Section 2.6). Constraints (6.3) are the configuration constraints that state that a run arc must be used if and only if the corresponding arc in the configuration network is used.



(a) Train A (b) Train B

Figure 6.8: Fractional solutions for the clique based model. The left picture shows the development of the fractional solution of train A, the right of train B. Each "column" of the picture corresponds to one iteration of the bundle method. Each "row" corresponds to one time step from $t = 1$ at the bottom to $t = 11$ at the top. One can see that train A (the long distance train) remains at time step $t = 1$ whereas train B is moved quickly to time step $t = 11$ after 7 iterations. Note that in this example all fractional solutions are in fact integral.



(a) Train A

(b) Train B

Figure 6.9: Fractional solutions for the configuration based model. The upper picture shows the development of the fractional solution of train A, the lower of train B. Each "column" of the picture corresponds to one iteration of the bundle method. Each "row" corresponds to one time step from $t = 1$ at the bottom to $t = 11$ at the top. The grey value corresponds to the fractional value of a variable, white means 0 and black means 1. One can see that train A (the long distance train) remains at time step $t = 1$ whereas train B is moved very slowly to time step $t = 11$. In each iteration the bundle method adjusts the costs of the arcs only slightly building the fractional solution as convex combination of the paths returned by the oracle. It requires several iterations until the Lagrange-multipliers are adjusted so that train B tries to use the run at $t = 2$ with a value of (almost) one. Then the fractional solution of B is moved slowly to $t = 3$ and so on until it reaches $t = 11$ after over 350 iterations.

In the first iteration the configuration constraint for one of the trains at time $t = 1$ is violated. This increases the costs *of only this specific run arc of the train.* This repeats until the costs of this arc is increased enough so that the train will try the next possible run arc at $t = 2$. Then the configuration constraint at $t = 2$ will be violated (because the other train still runs at $t = 1$) and so on. The difference to the clique based model is that in each iteration only one constraint will be violated and the subgradient of this violated constraint will lead to a change of the dual multipliers of only this one constraint. This causes the bundle method to require many iterations until the multipliers of all run arcs of the time steps $t \in \{1, \ldots, 10\}$ of train B are increased so that B will use the run arc at $t = 11$. The development of the fractional solution of the configuration based model is shown in Figure 6.9. This figure illustrates that the described effect is quite dramatic: the number of iterations required by the bundle method is 50 times larger than for the clique based model (the clique based model required 7 iterations, the configuration based model required 350).

Figure 6.10 shows the development of the objective value (*i. e.*, the lower bounds determined by the Lagrangian relaxation) for the RM-8 instance for both models. The convergence speed of the clique based model is far better than the speed of the configuration

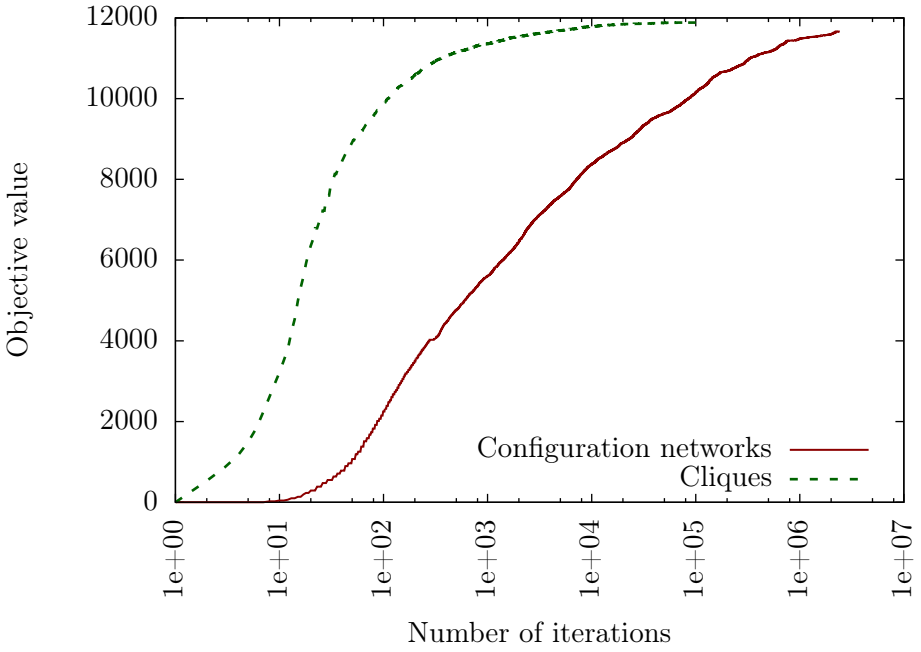


Figure 6.10: The plot shows the objective value (*i. e.*, the lower bound) after a certain number of iterations of the bundle method. The bound computed by the clique based model converges much faster (few minutes) than one computed by the configuration based model (several hours).

based model.

Note that this effect is not specific to bundle methods but to first order methods in general. In each iteration a first order method only gets information about the violated constraints and the corresponding subgradient. The subgradient of a violated clique constraint will affect *all* arcs covered by the clique, whereas the subgradient of a violated configuration constraint will only affect one single run arc. This effect will get even worse if the discretisation step size is reduced (in practice a discretisation step size of ten seconds is sometimes required for railway planning, *e. g.* for planning problems in big stations). The behaviour of the clique model will remain relatively stable because the clique constraint will still affect a whole time interval. However, the configuration based model will suffer because the number of configuration constraints increases a lot but each single constraint will still affect only one specific run arc.

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] R. K. Ahuja, J. B. Orlin, S. Pallottino, and M. G. Scutellà. "Dynamic shortest paths minimizing travel times and costs". In: *Networks* 41 (2003), p. 205.

[2] J. Aronson. "A survey of dynamic network flows". In: *Annals of Operations Research* 20 (1 1989), pp. 1–66. ISSN: 0254-5330. URL: http://dx.doi.org/10.1007/BF02216922.

[3] F. Babonneau, O. Du Merle, and J.-P. Vial. "Solving large-scale linear multicommodity flow problems with an active set strategy and proximal-ACCPM." In: *Oper. Res.* 54.1 (2006), pp. 184–197.

[4] A. Belloni and C. Sagastizábal. "Dynamic bundle methods". In: *Mathematical Programming* 120 (2 2009), pp. 289–311. ISSN: 0025-5610. URL: http://dx.doi.org/10.1007/s10107-008-0215-z.

[5] A. Bley, T. Koch, and R. Wessaly. "Large-scale hierarchical networks: how to compute an optimal architecture?" In: *Telecommunications Network Strategy and Planning Symposium. NETWORKS 2004, 11th International.* June 2004, pp. 429–434.

[6] J. F. Bonnans, J. C. Gilbert, C. Lemaréchal, and C. A. Sagastizábal. *Numerical Optimization.* Springer, 2003.

[7] R. Borndörfer, M. Grötschel, S. Lukac, K. Mitusch, T. Schlechte, S. Schultz, and A. Tanner. "An Auctioning Approach to Railway Slot Allocation". In: *Competition and Regulation in Network Industries* 1.2 (2006), pp. 163–196. URL: http://opus.kobv.de/zib/volltexte/2005/878.

[8] R. Borndörfer, A. Löbel, and S. Weider. "A Bundle Method for Integrated Multi-Depot Vehicle and Duty Scheduling in Public Transit". In: *Computer-aided Systems in Public Transport.* Ed. by M. Hickman, P. Mirchandani, and S. Voß. Vol. 600. Lecture Notes in Economics and Mathematical Systems. Springer Berlin Heidelberg, 2008, pp. 3–24. ISBN: 978-3-540-73311-9. URL: http://dx.doi.org/10.1007/978-3-540-73312-6_1.

[9] R. Borndörfer and T. Schlechte. "Models for Railway Track Allocation". In: *ATMOS 2007.* Ed. by C. Liebchen, R. K. Ahuja, and J. A. Mesa. Dagstuhl, Germany: IBFI, Schloss Dagstuhl, Germany, 2007. ISBN: 978-3-939897-04-0. URL: http://drops.dagstuhl.de/opus/volltexte/2007/1170.

[10] S. Boyd, L. Xiao, A. Mutapcic, and J. Mattingley. "Notes on Decomposition Methods". In: *Notes* D.2006 (2007), pp. 1–36. URL: http://www.core.org.cn/mirrors/Stanford/stanford/see.stanford.edu/materials/lsocoee364b/08-decomposition_notes.pdf.

[11] U. Brännlund, P. O. Lindberg, A. Nõu, and J. E. Nilsson. "Railway Timetabling Using Lagrangian Relaxation". In: *Transportation Science* 32.4 (1998), pp. 358–369. ISSN: 1526-5447.

[12] O. Briant, C. Lemaréchal, P. Meurdesoif, S. Michel, N. Perrot, and F. Vanderbeck. "Comparison of bundle and classical column generation". In: *Mathematical Programming* 113 (2 2008), pp. 299–344. ISSN: 0025-5610. URL: http://dx.doi.org/10.1007/s10107-006-0079-z.

[13] V. Cacchiani, A. Caprara, and P. Toth. "A column generation approach to train timetabling on a corridor". In: *4OR: A Quarterly Journal of Operations Research* 6.2 (June 2008), pp. 125–142. ISSN: 1619-4500 (Print) 1614-2411 (Online). URL: http://www.springerlink.com/content/c638656380058577/.

[14] V. Cacchiani, A. Caprara, and P. Toth. "Scheduling extra freight trains on railway networks". In: *Transportation Research Part B: Methodological* 44.2 (2010), pp. 215–231. ISSN: 0191-2615. URL: http://www.sciencedirect.com/science/article/pii/S0191261509000812.

[15] V. Cacchiani and P. Toth. "Nominal and robust train timetabling problems". In: *European Journal of Operational Research* 219.3 (2012), pp. 727–737. ISSN: 0377-2217. URL: http://sciencedirect.com/science/article/pii/S0377221711009908.

[16] G. C. Caimi. "Algorithmic decision support for train scheduling in a large and highly utilised railway network". PhD thesis. ETH Zurich, 2009.

[17] G. C. Caimi, M. Fuchsberger, M. Laumanns, and K. Schüpbach. "09. Periodic Railway Timetabling with Event Flexibility". In: *ATMOS 2007 - 7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems*. Ed. by C. Liebchen, R. K. Ahuja, and J. A. Mesa. Dagstuhl, Germany: Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2007. ISBN: 978-3-939897-04-0. URL: http://drops.dagstuhl.de/opus/volltexte/2007/1173.

[18] G. C. Caimi, M. Fuchsberger, M. Laumanns, K. Schüpbach, and S. Wörner. "The Periodic Service Intention as a Conceptual Framework for Generating Timetables with Partial Periodicity". In: *ISROR Proceedings, 2009*. 2009. URL: http://www.ifor.math.ethz.ch/publications/2009_caimi_partialperiodic_railzurich.pdf.

[19] A. Caprara, M. Fischetti, and P. Toth. "Modeling and Solving the Train Timetabling Problem". In: *Oper. Res.* 50.5 (2002), pp. 851–861. ISSN: 0030-364X.

[20] A. Caprara, L. Kroon, M. Monaci, M. Peeters, and P. Toth. "Passenger Railway Optimization". In: *Handbooks in Operations Research and Management Science*. Ed. by C. Barnhart and G. Laporte. Vol. 14. Elsevier, 2007. Chap. 3, pp. 129–187. URL: http://www.sciencedirect.com/science/article/pii/S0927050706140037.

[21] A. Caprara, M. Monaci, P. Toth, and P. L. Guida. "A Lagrangian heuristic algorithm for a real-world train timetabling problem". In: *Discrete Appl. Math.* 154.5 (2006), pp. 738–753. ISSN: 0166-218X.

[22] M. Carey and D. Lockwood. "A Model, Algorithms and Strategy for Train Pathing". In: *Journal of the Operational Research Society* 46.8 (Aug. 1995), pp. 988–1005. URL: http://www.jstor.org/stable/3009909.

[23] T. G. Crainic, A. Frangioni, and B. Gendron. "Bundle-based relaxation methods for multicommodity capacitated fixed charge network design". In: *Discrete Applied Mathematics* 112.1-3 (2001). Combinatorial Optimization Symposium, Selected Papers, pp. 73–99. ISSN: 0166-218X. URL: http://www.sciencedirect.com/science/article/pii/S0166218X00003103.

[24] G. B. Dantzig and P. Wolfe. "Decomposition Principle for Linear Programs". In: *Operations Research* 8.1 (1960), pp. 101–111. URL: http://dx.doi.org/10.2307/167547.

[25] G. Dantzig, R. Fulkerson, and S. Johnson. "Solution of a large-scale traveling-salesman problem". In: *Operations Research* 2 (1954), pp. 393–410.

[26] D. Delling and D. Wagner. "Landmark-Based Routing in Dynamic Graphs". In: *Experimental Algorithms*. Ed. by C. Demetrescu. Vol. 4525. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2007, pp. 52–65. ISBN: 978-3-540-72844-3. URL: http://dx.doi.org/10.1007/978-3-540-72845-0_5.

[27] G. Desaulniers, J. Desrosiers, and M. M. Solomon, eds. *Column generation*. Springer US, 2005.

[28] J. Desrosiers and M. Lübbecke. "A Primer in Column Generation". In: *Column Generation*. Ed. by G. Desaulniers, J. Desrosiers, and M. M. Solomon. Springer US, 2005. Chap. 1, pp. 1–32. ISBN: 978-0-387-25485-2. URL: http://dx.doi.org/10.1007/0-387-25486-2_1.

[29] R. Diestel. *Graph Theory*. Fourth. Vol. 173. Graduate Texts in Mathematics. Springer-Verlag, Heidelberg, 2010.

[30] E. W. Dijkstra. "A note on two problems in connexion with graphs". In: *Numerische Mathematik* 1 (1959), pp. 269–271. URL: http://jmvidal.cse.sc.edu/library/dijkstra59a.pdf.

[31] S. Feltenmark and K. C. Kiwiel. "Dual applications of proximal bundle methods, including Lagrangian relaxation of nonconvex problems". In: *SIAM Journal on Optimization* 10.3 (2000), pp. 697–721.

[32] M. C. Ferris and O. L. Mangasarian. "Parallel Variable Distribution". In: *SIAM Journal on Optimization* 4.4 (1994), pp. 815–832. URL: http://link.aip.org/link/?SJE/4/815/1.

[33] F. Fischer and C. Helmberg. *A Parallel Bundle Method for Asynchronous Subspace Optimization in Lagrangian Relaxation*. Preprint 2012-02. Submitted to SIAM Journal on Optimization. D-09107 Chemnitz, Germany: Technische Universität Chemnitz, Fakultät für Mathematik, Feb. 2012. URL: http://www.tu-chemnitz.de/mathematik/preprint/2012/PREPRINT_02.pdf.

[34] F. Fischer and C. Helmberg. "Dynamic Graph Generation and Dynamic Rolling Horizon Techniques in Large Scale Train Timetabling". In: *Proceedings of the 10th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*. Ed. by T. Erlebach and M. Lübbecke. Vol. 14. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010, pp. 45–60. ISBN: 978-3-939897-20-0. URL: http://drops.dagstuhl.de/opus/volltexte/2010/2749.

[35] F. Fischer and C. Helmberg. "Dynamic graph generation for the shortest path problem in time expanded networks". In: *Mathematical Programming* (2012), pp. 1–41. ISSN: 0025-5610. URL: http://dx.doi.org/10.1007/s10107-012-0610-3.

[36] F. Fischer, C. Helmberg, J. Janßen, and B. Krostitz. "Towards Solving Very Large Scale Train Timetabling Problems by Lagrangian Relaxation". In: *ATMOS 2008 - 8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems*. Ed. by M. Fischetti and P. Widmayer. Dagstuhl, Germany: Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2008. URL: http://drops.dagstuhl.de/opus/volltexte/2008/1585.

[37] M. L. Fisher. "The Lagrangian Relaxation Method for Solving Integer Programming Problems". In: *Manage. Sci.* 50.12 Supplement (Dec. 2004), pp. 1861–1871. ISSN: 0025-1909. URL: http://dl.acm.org/citation.cfm?id=1245920.1245938.

[38] M. Fukushima. "Parallel Variable Transformation in Unconstrained Optimization". In: *SIAM J. on Optimization* 8.3 (Mar. 1998), pp. 658–672. ISSN: 1052-6234. URL: http://dl.acm.org/citation.cfm?id=588907.589327.

[39] S. Gawiejnowicz. *Time-Dependent Scheduling*. 1st ed. Berlin: Springer, 2008. ISBN: 3540694455, 9783540694458.

[40] A. V. Goldberg. "Point-to-Point Shortest Path Algorithms with Preprocessing". In: *SOFSEM 2007: Theory and Practice of Computer Science*. Ed. by J. Leeuwen, G. Italiano, W. Hoek, C. Meinel, H. Sack, and F. Plášil. Vol. 4362. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, pp. 88–102. ISBN: 978-3-540-69506-6. URL: http://dx.doi.org/10.1007/978-3-540-69507-3_6.

[41] A. V. Goldberg, H. Kaplan, and R. F. Werneck. "Reach for A^*: Shortest Path Algorithms with Preprocessing." In: *The shortest path problem. Ninth DIMACS implementation challenge, Piscataway, NJ, USA, November 13–14, 2006. Proceedings*. Providence, RI: American Mathematical Society (AMS), 2009, pp. 93–139. ISBN: 978-0-8218-4383-3.

[42] D. Goldfarb and S. Ma. *Fast Alternating Linearization Methods for Minimizing the Sum of Two Convex Functions*. Tech. rep. arXiv:0912.4571. Dec. 2009. URL: http://www.optimization-online.org/DB_HTML/2009/12/2502.html.

[43] D. Goldfarb and S. Ma. *Fast Multiple Splitting Algorithms for Convex Optimization*. Tech. rep. arXiv:0912.4570. Dec. 2009. URL: http://www.optimization-online.org/DB_HTML/2009/12/2501.html.

[44] J. Gondzio and A. Grothey. "Parallel interior-point solver for structured quadratic programs: Application to financial planning problems". In: *Annals of Operations Research* 152.1 (2007), pp. 319–339. URL: http://www.ingentaconnect.com/content/klu/anor/2007/00000152/00000001/00000139.

[45] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan. "Optimization and approximation in deterministic sequencing and scheduling: a survey". In: *Ann. Discrete Math.* 4 (1979), pp. 287–326. URL: http://www.math.ucsd.edu/~ronspubs/79_03_scheduling_survey.pdf.

[46] H. Y. Gu. "Computation of approximate α-points for large scale single machine scheduling problem". In: *Computers & Operations Research* 35.10 (2008). Part Special Issue: Search-based Software Engineering, pp. 3262–3275. ISSN: 0305-0548. URL: http://sciencedirect.com/science/article/pii/S0305054807000561.

[47] S. Hanafi, C. Sterle, A. Ushakov, and I. Vasilyev. "A parallel subgradient algorithm for Lagrangean dual function of the p-median problem". In: *Studia Informatica Universalis* 9.3 (2011), pp. 105–124. URL: http://studia.complexica.net/Art/RI090309.pdf.

[48] P. Hart, N. Nilsson, and B. Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". In: *Systems Science and Cybernetics, IEEE Transactions on* 4.2 (July 1968), pp. 100–107. ISSN: 0536-1567.

[49] B. He, T. Min, and X. Yuan. *A splitting method for separate convex programming with linking linear constraints*. Tech. rep. 2008. URL: http://www.optimization-online.org/DB_HTML/2010/06/2665.html.

[50] C. Helmberg and K. Kiwiel. "A spectral bundle method with bounds". In: *Mathematical Programming* 93 (2 2002), pp. 173–194. ISSN: 0025-5610. URL: http://dx.doi.org/10.1007/s101070100270.

[51] C. Helmberg. "A Cutting Plane Algorithm for Large Scale Semidefinite Relaxations". In: *The Sharpest Cut*. Ed. by M. Grötschel. Vol. 4. MPS-SIAM Series on Optimization. SIAM/MPS, 2004, pp. 233–256. ISBN: 0-89871-552-0.

[52] C. Helmberg. *ConicBundle 0.3.11*. Fakultät für Mathematik, Technische Universität Chemnitz. 2012. URL: http://www.tu-chemnitz.de/~helmberg/ConicBundle.

[53] C. Helmberg and S. Röhl. "A Case Study of Joint Online Truck Scheduling and Inventory Management for Multiple Warehouses". In: *Operations Research* 55.4 (July 2007), pp. 733–752.

[54] A. Higgins, E. Kozan, and L. Ferreira. "Optimal scheduling of trains on a single line track". In: *Transportation Research Part B: Methodological* 30.2 (Apr. 1996), pp. 147–161. URL: http://ideas.repec.org/a/eee/transb/v30y1996i2p147-161.html.

[55] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms I & II*. Vol. 305, 306. Grundlehren der mathematischen Wissenschaften. Berlin, Heidelberg: Springer, 1993.

[56] *IBM ILOG CPLEX 12.1, User's Manual for CPLEX, 2009*. URL: http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.

[57] N. Kamiyama, N. Katoh, and A. Takizawa. "An efficient algorithm for the evacuation problem in a certain class of networks with uniform path-lengths". In: *Discrete Appl. Math.* 157.17 (Oct. 2009), pp. 3665–3677. ISSN: 0166-218X. URL: http://dx.doi.org/10.1016/j.dam.2009.04.007.

[58] K. Kiwiel. "Approximations in proximal bundle methods and decomposition of convex programs". In: *Journal of Optimization Theory and Applications* 84 (3 1995), pp. 529–548. ISSN: 0022-3239. URL: http://dx.doi.org/10.1007/BF02191984.

[59] K. Kiwiel and P. Lindberg. "Parallel subgradient methods for convex optimization". In: *Inherently Parallel Algorithms in Feasibility and Optimization and their Applications*. Ed. by Y. C. Dan Butnariu and S. Reich. Vol. 8. Studies in Computational Mathematics. Elsevier, 2001, pp. 335–344. URL: http://www.sciencedirect.com/science/article/pii/S1570579X01800203.

[60] G. A. Klunder and H. N. Post. "The shortest path problem on large-scale real-road networks". In: *Networks* 48.4 (Dec. 2006), pp. 182–194. ISSN: 0028-3045. URL: http://dx.doi.org/10.1002/net.v48:4.

[61] B. Kotnyek. *An annotated overview of dynamic network flows*. Tech. rep. RR-4936. INRIA, Sept. 2003. URL: http://hal.inria.fr/inria-00071643.

[62] L. Kroon, D. Huisman, E. Abbink, P.-J. Fioole, M. Fischetti, G. Maróti, A. Schrijver, A. Steenbeek, and R. Ybema. "The New Dutch Timetable: The OR Revolution". In: *Interfaces* 39.1 (Jan. 2009), pp. 6–17. ISSN: 0092-2102. URL: http://dx.doi.org/10.1287/inte.1080.0409.

[63] C. Lemaréchal and A. Renaud. "A geometric study of duality gaps, with applications". In: *Mathematical Programming* 90.3 (2001), pp. 399–427. ISSN: 0025-5610. URL: http://dx.doi.org/10.1007/PL00011429.

[64] C. Lemaréchal. "Lagrangian Relaxation". In: *Computational Combinatorial Optimization*. Ed. by M. Jünger and D. Naddef. Vol. 2241. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2001, pp. 112–156. ISBN: 978-3-540-42877-0. URL: http://dx.doi.org/10.1007/3-540-45586-8_4.

[65] C. Lemaréchal. "The omnipresence of Lagrange". In: *Annals of Operations Research* 153.1 (2007), pp. 9–27. ISSN: 0254-5330. URL: http://dx.doi.org/10.1007/s10479-007-0169-1.

[66] C. Lemaréchal, A. Nemirovskii, and Y. Nesterov. "New variants of bundle methods". In: *Mathematical Programming* 69 (1-3 1995), pp. 111–147. ISSN: 0025-5610. URL: http://dx.doi.org/10.1007/BF01585555.

[67] C. Lemaréchal, A. Ouorou, and G. Petrou. "A bundle-type algorithm for routing in telecommunication data networks". In: *Computational Optimization and Applications* 44 (3 2009), pp. 385–409. ISSN: 0926-6003. URL: http://dx.doi.org/10.1007/s10589-007-9160-7.

[68] C. Liebchen. "Periodic Timetable Optimization in Public Transport". PhD thesis. Technical University Berlin, 2006.

[69] C. Liebchen and R. H. Möhring. "The Modeling Power of the Periodic Event Scheduling Problem: Railway Timetables – and Beyond". In: *Algorithmic Methods for Railway Optimization*. Ed. by F. Geraets, L. Kroon, A. Schoebel, D. Wagner, and C. D. Zaroliagis. Vol. 4359. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, pp. 3–40. ISBN: 978-3-540-74245-6. URL: http://dx.doi.org/10.1007/978-3-540-74247-0_1.

[70] M. E. Lübbecke and J. Desrosiers. "Selected Topics in Column Generation." In: *Operations Research* 53.6 (2005), pp. 1007–1023. ISSN: 0030364X. URL: http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=19323613&site=ehost-live.

[71] R. M. Lusby, J. Larsen, M. Ehrgott, and D. Ryan. "Railway track allocation: models and methods". In: *OR Spectr.* 33.4 (Oct. 2011), pp. 843–883. ISSN: 0171-6468. URL: http://dx.doi.org/10.1007/s00291-009-0189-0.

[72] D. Medhi. "Parallel bundle-based decomposition for large-scale structured mathematical programming problems". In: *Ann. Oper. Res.* 22.1-4 (July 1990), pp. 101–127. ISSN: 0254-5330. URL: http://portal.acm.org/citation.cfm?id=85598.85620.

[73] F. Meng, G. Zhao, M. Goh, and R. D. Souza. "Lagrangian-Dual Functions and Moreau–Yosida Regularization". In: *SIAM Journal on Optimization* 19.1 (2008), pp. 39–61. URL: http://link.aip.org/link/?SJE/19/39/1.

[74] A. J. Mezger and K. C. de Almeida. "Short term hydrothermal scheduling with bilateral transactions via bundle method". In: *International Journal of Electrical Power & Energy Systems* 29.5 (2007), pp. 387–396. ISSN: 0142-0615. URL: http://www.sciencedirect.com/science/article/pii/S014206150600189X.

[75] M. Nayakkankuppam. "Solving large-scale semidefinite programs in parallel". In: *Mathematical Programming* 109.2 (2007), pp. 477–504. ISSN: 0025-5610. URL: http://dx.doi.org/10.1007/s10107-006-0032-1.

[76] I. Necoara and J. Suykens. "Interior-Point Lagrangian Decomposition Method for Separable Convex Optimization". In: *Journal of Optimization Theory and Applications* 143.3 (2009), pp. 567–588. ISSN: 0022-3239. URL: http://dx.doi.org/10.1007/s10957-009-9566-8.

[77] A. Nedić, D. Bertsekas, and V. Borkar. "Distributed asynchronous incremental subgradient methods". In: *Inherently Parallel Algorithms in Feasibility and Optimization and their Applications*. Ed. by Y. C. Dan Butnariu and S. Reich. Vol. 8. Studies in Computational Mathematics. Elsevier, 2001, pp. 381–407. URL: http://www.sciencedirect.com/science/article/pii/S1570579X01800239.

[78] Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Academic, 2004.

[79] W. Oliveira, C. Sagastizábal, and S. Scheimberg. "Inexact Bundle Methods for Two-Stage Stochastic Programming". In: *SIAM J. Optim.* 21.2 (2011), pp. 517–544.

[80] L. W. P. Peeters. "Cyclic Railway Timetable Optimization". PhD thesis. Erasmus Research Institute of Management, June 2003. URL: http://hdl.handle.net/1765/429.

[81] M. Potamias, F. Bonchi, C. Castillo, and A. Gionis. "Fast shortest path distance estimation in large networks". In: *Proceedings of the 18th ACM conference on Information and knowledge management*. CIKM '09. New York, NY, USA: ACM, 2009, pp. 867–876. ISBN: 978-1-60558-512-3. URL: http://doi.acm.org/10.1145/1645953.1646063.

[82] P. Richtárik and M. Takáč. "Parallel Coordinate Descent Methods for Big Data Optimization". In: *ArXiv e-prints* (Dec. 2012). arXiv: 1212.0873 [math.OC]. URL: http://arxiv.org/abs/1212.0873.

[83] C. Roos, T. Terlaky, and J.-P. Vial. *Interior point methods for linear optimization*. 2nd. Springer, 2006.

[84] C. A. Sagastizábal and M. V. Solodov. "Parallel Variable Distribution for Constrained Optimization". In: *Comput. Optim. Appl.* 22.1 (Apr. 2002), pp. 111–131. ISSN: 0926-6003. URL: http://dl.acm.org/citation.cfm?id=584529.584535.

[85] C. Sagastizábal and M. Solodov. "Solving generation expansion planning problems with environmental constraints by a bundle method". In: *Computational Management Science* 9 (2 2012), pp. 163–182. ISSN: 1619-697X. URL: http://dx.doi.org/10.1007/s10287-012-0139-1.

[86] T. Schlechte. "Railway Track Allocation: Models and Algorithms". PhD thesis. Technische Universität Berlin, 2012. URL: http://opus.kobv.de/tuberlin/volltexte/2012/3427/pdf/schlechte_thomas.pdf.

[87] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 2000.

[88] R. Schultz. "Stochastic programming with integer variables". In: *Math. Programming* 97.1-2 (2003), pp. 285–309.

[89] P. Serafini and W. Ukovich. "A mathematical model for periodic scheduling problems". In: *SIAM J. Discret. Math.* 2.4 (1989), pp. 550–581. ISSN: 0895-4801.

[90] M. Skutella. "An Introduction to Network Flows over Time". In: *Research Trends in Combinatorial Optimization*. Springer-Verlag, 2009, pp. 451–482. URL: http://dx.doi.org/10.1007/978-3-540-76796-1_21.

[91] B. Szpigel. "Optimal train scheduling on a single track railway". In: *Proceedings of IFORS Conference on Operational Research '72*. 1973, pp. 343–352.

[92] J. R. Tebboth. "A Computational Study of Dantzig-Wolfe Decomposition". PhD thesis. University of Buckingham, 2001. URL: http://eaton.math.rpi.edu/CourseMaterials/Spring08/JM6640/tebboth.pdf.

[93] R. J. Vanderbei. *Linear Programming: Foundations and Extensions*. 2nd. Springer, 2001.

[94] H. Yanagisawa. *Fast Shortest Path Computation for Solving the Multicommodity Flow Problem*. Tech. rep. RT0688. IBM Tokyo Research Laboratory, 2006.

[95] D. Zhang, P. Luh, and Y. Zhang. "A bundle method for hydrothermal scheduling". In: *Power Systems, IEEE Transactions on* 14.4 (Nov. 1999), pp. 1355–1361. ISSN: 0885-8950.

Theses

One successful solution approach for large scale combinatorial optimisation problems is Lagrangian relaxation. However, many instances of practical size lead to computationally intractable models. One of these problems is the *train timetabling problem* (TTP). In the thesis "Dynamic Graph Generation and an Asynchronous Parallel Bundle Method Motivated by Train Timetabling", motivated by a practical TTP from *Deutsche Bahn*, we develop two algorithmic techniques to improve the solution process for this kind of optimisation problems.

1. The TTP asks for a conflict free schedule of freight and passenger trains in a railway infrastructure network so that the delays of the passenger trains and the arrival times of freight trains are minimised. The route of each train is a fixed path in the infrastructure network.

2. We consider large scale instances from *Deutsche Bahn* of about 10% of the German railway network (about 2.500 trains in 6 hours, the whole German network contains more than 30.000 trains per day). All running times and headway times depend on the train types and on whether a train stops at a station or passes through a station without stopping.

3. We formulate two integer programming models for the TTP. Both are based on time expanded networks with linear coupling constraints to ensure the capacity restrictions in the infrastructure nodes and arcs. These types of models have proved to be the most successful ones for the TTP in the literature. One model ensures the headway restrictions using inequality constraints that forbid the simultaneous use of arcs that are in pairwise conflict. The second model describes conflict-free *configurations* of each infrastructure arc in an additional configuration network. A configuration corresponds to a selection of train runs that observe the headway constraints. The configuration networks are coupled with the train graphs so that only runs compatible with the configuration are allowed.

4. The basic solution approach is based on Lagrangian relaxation. The resulting large scale, non-smooth, convex optimisation problem is solved using a proximal bundle method. In each iteration the solution of one independent shortest path problem in each time expanded network is required.

5. Models based on time expanded networks get very large for big real world instances so that even the solution of the relaxation becomes computationally challenging. We develop two algorithmic approaches to improve the tractability of these models: a *Dynamic Graph Generation* technique and an *Asynchronous Parallel Bundle Method*.

6. Dynamic Graph Generation exploits the property of scheduling problems like the TTP to prefer early times (*e. g.*, reduce delays, early completion time, ...). During the solution process most trains tend to use only the early time steps in their respective time expanded networks.

7. The developed Dynamic Graph Generation technique focuses on the efficient solution of the shortest path problems in each time expanded network. It stores only small parts of the networks in memory. The algorithm ensures that this part is large enough to certify the correctness of the solved shortest path problem and, otherwise, dynamically adds further parts to the network. In particular, the algorithm guarantees that all subproblem evaluations are exact, *i. e.*, the use of Dynamic Graph Generation does not sacrifice any accuracy of the model or the solution algorithm.

8. Due to the automatic growing of the networks, Dynamic Graph Generation allows to deal with models that have an infinite time horizon under some weak assumptions. Although not required by our TTP models, Dynamic Graph Generation can be used with routing problems as well.

9. The implemented algorithm is very efficient. Its running time does not depend on the size of the time expanded networks but only (linearly) on the length of the train routes. Furthermore we prove that the size of the networks to be stored in memory exceeds the subnetwork that contains all shortest paths of all evaluations (which must be contained in the subnetwork) only by a constant number of time steps if no routing is involved.

10. Numerical experiments on large scale real world instances of the TTP show a reduction of the number of arcs to be stored in memory by a factor of 40.

11. We develop an asynchronous parallel bundle method for Lagrangian relaxation. A classical bundle method requires one solution of each subproblem in each iteration. However, many practical applications, *e. g.* the TTP, consist of many "loosely" coupled subproblems in the sense that each single coupling constraint interacts with only few subproblems. The asynchronous parallel bundle method exploits this structure by choosing a (small) subspace of violated constraints and optimising only this subspace and the subproblems that are coupled by these constraints by a proximal bundle method. Several subspaces with disjoint sets of coupled subproblems can be chosen simultaneously and are optimised in parallel. When a subspace problem has been optimised to a required precision, its computed solution is incorporated into the global data. There is no explicit synchronisation of the order in which new subspace problems are started or in which the global data is updated.

12. Optimising over one subspace problem reduces the conflicts in the associated constraints but may introduce conflicts with other constraints that are not part of the subspace. Dependencies that may endanger convergence of the algorithm are detected automatically and observed in future subspace selections.

13. We prove the convergence of the algorithm under reasonable assumptions. This includes the converge of appropriate subsequences of dual multipliers and primal aggregates to dual resp. primal optimal solutions of the Lagrangian relaxation problem.

14. The standard parallel bundle method considers a subproblem as interacting with a constraint if it contains at least one non-zero coefficient in that constraint. This might lead to relatively large subspaces and is a common structure in applications like the TTP. We develop an extended form of the algorithm that, in addition to the usual dependency detection between constraints, automatically detects which constraints are active (*i. e.*, restrictive) for which subproblem. Again convergence of the algorithm under reasonable assumptions is shown.

15. The extended parallel bundle method is implemented and tested with random test instances of the TTP. The numerical results show that our algorithm can reduce the number of required subproblem evaluations significantly compared with a classical proximal bundle method.

16. Our solution approach is tested on some real world instances of *Deutsche Bahn*. We use a heuristic to round the fractional solutions computed by the Lagrangian relaxation approach to feasible integer solutions in an iterative scheme. The computed timetables find feasible schedules with only small delays for 99% of all passenger trains and reduce the average arrival time of the freight trains by about one hour.

17. We compare the two models for the TTP based on clique constraints resp. configuration networks. It is known that configuration based formulations lead to stronger relaxations. However, we demonstrate that the clique based formulation behaves algorithmically much better than the configuration based formulation when used in a Lagrangian relaxation approach in conjunction with first order methods, *i. e.*, clique based formulations converge much faster.

Curriculum Vitae

Personal information

Name	Frank Fischer
Date of birth	July 22, 1981
Place of birth	Karl-Marx-Stadt, Germany
E-mail	`frank.fischer@mathematik.tu-chemnitz.de`
Family status	Married, no children
Citizenship	German

Scientific experience

since 09/2007	Research assistant at Chemnitz University of Technology, Department of Mathematics, Professorship of Algorithmic and Discrete Mathematics, in the research projects OVERSYS and KOSMOS supported by the *Bundesministerium für Bildung und Forschung*
03/2003–09/2006	Student assistant at Chemnitz University of Technology, Department of Mathematics
10/2002–02/2003	Student assistant at Chemnitz University of Technology, Department of Computer Science

Education

9/2001–07/2007	Studies in Computer Science, diploma in July 2007, diploma thesis: Effizientes Finden kleiner unabhängiger Mengen in zufälligen Graphen mit gegebener erwarteter Gradsequenz
9/2004–07/2007	Studies in Mathematics
06/2000	Abitur at Gottfried-Leibniz-Gymnasium Chemnitz

Publications

[1] A. Fischer and F. Fischer. *An extended approach for lifting clique tree inequalities.* Preprint 2012-13. Submitted to Journal of Combinatorial Optimization. D-09107 Chemnitz, Germany: Fakultät für Mathematik, Technische Universität Chemnitz, Nov. 2012.

[2] F. Fischer and C. Helmberg. *A Parallel Bundle Method for Asynchronous Subspace Optimization in Lagrangian Relaxation*. Preprint 2012-02. Submitted to SIAM Journal on Optimization. D-09107 Chemnitz, Germany: Technische Universität Chemnitz, Fakultät für Mathematik, Feb. 2012. URL: http://www.tu-chemnitz.de/mathematik/preprint/2012/PREPRINT_02.pdf.

[3] F. Fischer and C. Helmberg. "Dynamic Graph Generation and Dynamic Rolling Horizon Techniques in Large Scale Train Timetabling". In: *Proceedings of the 10th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*. Ed. by T. Erlebach and M. Lübbecke. Vol. 14. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010, pp. 45–60. ISBN: 978-3-939897-20-0. URL: http://drops.dagstuhl.de/opus/volltexte/2010/2749.

[4] F. Fischer and C. Helmberg. *Dynamic Graph Generation for Large Scale Operational Train Timetabling*. Preprint 2011-10. D-09107 Chemnitz, Germany: Technische Universität Chemnitz, Fakultät für Mathematik, 2011. URL: http://www.tu-chemnitz.de/mathematik/preprint/2011/PREPRINT_10.pdf.

[5] F. Fischer and C. Helmberg. "Dynamic graph generation for the shortest path problem in time expanded networks". In: *Mathematical Programming* (2012), pp. 1–41. ISSN: 0025-5610. URL: http://dx.doi.org/10.1007/s10107-012-0610-3.

[6] F. Fischer, C. Helmberg, J. Janßen, and B. Krostitz. "Towards Solving Very Large Scale Train Timetabling Problems by Lagrangian Relaxation". In: *ATMOS 2008 - 8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems*. Ed. by M. Fischetti and P. Widmayer. Dagstuhl, Germany: Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2008. URL: http://drops.dagstuhl.de/opus/volltexte/2008/1585.

[7] F. Fischer, G. Jäger, A. Lau, and P. Molitor. *Complexity and Algorithms for the Traveling Salesman Problem and the Assignment Problem of Second Order*. Preprint 2009-16. D-09107 Chemnitz, Germany: Technische Universität Chemnitz, Fakultät für Mathematik, 2009. URL: http://www.tu-chemnitz.de/~lauan/downloads/PR09-16.pdf.

[8] S. Krönert, A. Fischer, F. Fischer, and U. Götze. "Wirtschaftliche Analyse von Handlungsalternativen am Beispiel der energiesensitiven Koordination von Robotern in getakteten Fertigungsstraßen". In: *Energy Related Balancing and Evaluation in Production Engineering – Methods and Examples. Proceedings of the second workshop of the cross-sectional group "Energy-related technologic and economic evaluation" of the Cluster of Excellence eniPROD*. Ed. by R. Neugebauer, U. Götze, and W.-G. Drossel. Verlag Wissenschaftliche Scripten, 2013, pp. 213–230.

Erklärungen gemäß §6 der Promotionsordnung

Hiermit erkläre ich an Eides Statt, dass ich die von mir eingereichte Arbeit

„Dynamic Graph Generation and an
Asynchronous Parallel Bundle Method
Motivated by Train Timetabling“

- selbständig und nur unter Benutzung der in der Arbeit angegebenen Quellen und Hilfsmittel angefertigt habe,
- in dieser oder ähnlicher Form an keiner anderen Stelle zum Zwecke eines Promotionsverfahrens vorgelegt habe.

Hiermit erkläre ich an Eides Statt, dass ich keine weiteren Promotionsverfahren bei anderen Stellen beantragt hatte bzw. beantragt habe.

Chemnitz, den 02.04.2013 Frank Fischer